JPL PUBLICATION 83-17

Some Practical Universal Noiseless Coding Techniques, Part II

Robert F. Rice
Jun-Ji Lee

March 1, 1983

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL PUBLICATION 83-17

Some Practical Universal Noiseless Coding Techniques, Part II

Robert F. Rice
Jun-Ji Lee

March 1, 1983

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

ACKNOWLEDGEMENT

The authors wish to acknowledge the helpful suggestions and insight into Voyager processing constraints provided by Don Acord and Richard J. Rice. They would also like to thank Marie Curry and Virgil Austin for their competent assistance in preparing this report.

The research described in this paper was carried out by the Information Processing Research Group of the Jet Propulsion Laboratory, California Institute of Technology.

ABSTRACT

This paper is an extension of earlier work (Part I) which provided practical adaptive techniques for the **efficient** noiseless coding of a broad class of data sources characterized by only partially known and varying statistics (JPL Publication 79-22). The results here, while still claiming such general applicability, focus primarily on the noiseless coding of image data. A fairly complete and self-contained treatment is provided. Particular emphasis is given to the requirements of the forthcoming Voyager II encounters of Uranus and Neptune. Performance evaluations are supported both graphically and pictorially.

Expanded definitions of the algorithms in Part I yield a computationally improved set of options for applications requiring efficient performance at entropies above 4 bits/sample. These expanded definitions include as an important subset, a somewhat less efficient but extremely simple "FAST" compressor which will be used at the Voyager Uranus encounter. Additionally, options are provided which enhance performance when atypical data spikes may be present.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Figures

Tables

I. INTRODUCTION

References 1-3 provided the development and analysis of some practical adaptive techniques for the efficient noiseless coding of a broad class of data sources. Specifically, these algorithms were developed for efficiently coding discrete memoryless sources which have known symbol probability ordering but unknown probability values. A general applicability of these algorithms to solving practical problems is obtained because most real data sources can be simply transformed into this form by appropriate reversible preprocessing. Application to image data compression is a particularly important and straightforward example, having motivated most of this earlier work.[4]-[8]

This paper also derives its motivation from an image data compression problem arising from the combination of severe limitations in both on-board processing and available data rate of the Voyager II spacecraft which will encounter Uranus in 1986 and hopefully, Neptune in 1989. In addition to providing generalizations of the source independent algorithms of Refs. 1-3, this paper explicitly deals with the preprocessing requirements of monochrome image data sources, yielding a fairly complete treatment of noiseless image compression. The techniques presented span a broad range of performance and complexity. New results include the definition and evaluation of:

a) an extremely simple "FAST" compressor for the Uranus encounter;
b) algorithms which maintain efficient performance while reducing computation;
c) algorithms which improve performance;
d) algorithms to deal with atypical data spikes.

The material presented here is mostly self-contained although the reader may wish to review the additional concepts and background supplied by the references (e.g., explicit coding examples in Ref. 1). The notational convention and method of presentation developed in Ref. 1 will be followed although it will be reviewed as needed here. Although considerable attention is given to the Voyager problem, the reader should keep in mind the general applicability of these techniques to other imaging and non-imaging problems.

BACKGROUND AND REVIEW

Discrete data sources arising from practical problems are generally characterized by only partially known and varying statistics. As just noted, the algorithms in Refs. 1-3 were developed for efficiently coding discrete memoryless sources which have known symbol probability ordering but unknown probability values. A general applicability of these algorithms to solving practical problems is obtained because most real data sources can be simply transformed into this form by appropriate **reversible** preprocessing. The latter operations include those that first remove correlation (memory) to produce samples which are approximately independent and then relabels these into the integers 0, 1, 2, ... such that a smaller integer is more likely to occur than a larger one. That is, if p_i denotes the probability of integer i, then the desired condition resulting from preprocessing is

$$p_0 \geq p_1 \geq p_2 \geq \cdots \tag{1}$$

The reversible preprocessing step yielding (1) is followed by a mapping of the integers into variable length codewords. Compression is obtained on an average basis by using shorter codewords for the most frequently occurring integers (by (1), the smaller ones) and longer codewords for the less likely integers (the larger ones). It is the mechanism for assignment of codewords that Refs. 1-3 primarily address. Here, we will in addition, treat the reversible preprocessing function for image data sources.

Image Preprocessor

In some problems a preprocessor may need to be adaptive to ensure that the decorrelation process is maintained and that condition (1) remains well approximated. But for the imaging problem, such as Voyager's, the preprocessor can usually be fixed and quite simple.[4],[6],[7]

The first step in any application is to fully utilize any source memory or a priori information. For imaging this amounts to the generation of a sequence of "differences" from **predicted** picture element (pixel) values. The better that prediction process performs, the less uncertainty there is in the sequence of output symbols. This itself leads to fewer bits in the coding process to follow. However, using any more than the immediately adjacent pixels by a predictor offers negligible improvement in performance.

Thus practical one and two-dimensional predictors reduce to simple functions of the pixels immediately before and/or above a predicted pixel. Letting $x_{i,j}$ denote the jth pixel value on the ith line, the candidate practical predictors, $\hat{x}_{i,j}$, for $x_{i,j}$ are summarized in Table 1.†

Table 1. Practical Image Predictors.

ONE-DIMENSIONAL	TWO-DIMENSIONAL *
$\hat{x}_{i,j} = x_{i,j-1}$	$\hat{x}_{i,j} = \frac{x_{i,j-1} + x_{i-1,j}}{2}$
* use roundoff or truncation	

The approximately memoryless error signal, henceforth denoted as Δ, from either predictor is consistently distributed in a unimodal fashion about zero for virtually any scene activity. This leads to the condition

$$\Pr[\Delta = 0] \geq \Pr[\Delta = +1] \geq \Pr[\Delta = -1] \geq \Pr[\Delta = +2] \geq \ldots \qquad (2)$$

Not surprisingly, low detail scenes yield distributions more peaked around zero than high detail scenes because the prediction works better. Similarly, the distributions will generally be more peaked for the two-dimensional predictor although this is only significant on detailed scenes.

Because of the consistent **shape** of the error signal, Δ, the desired basic mapping into the integers, δ, such that the probability ordering of condition (1) is almost always well approximated, is given by Table 2 and expressed analytically as‡

† Observe that the performance of these very simple predictors would be noticeably affected by "sensor noise" (e.g., the familiar gain and offset variations of today's CCD cameras) but these variations can be considered negligible for the Voyager system.

‡ Observe that this statement remains true even if one or more pixel least significant bits are "shifted out" before data entry. The distributions of both Δ and δ would simply become more peaked at zero by this procedure.

Table 2. Basic Mapping of Δ into the Integers, δ.

Prediction Error Δ	Integer δ
0	0
+1	1
−1	2
+2	3
−2	4
+3	5
.	.
.	.
.	.

$$\delta = \begin{cases} 2\Delta - 1 & \text{if } \Delta > 0 \\ 2|\Delta| & \text{if } \Delta \leq 0. \end{cases} \tag{3}$$

This simple mapping is adequate for most imaging situations. However, it does not utilize a dynamic range constraint provided by predictor values $\hat{x}_{i,j}$ in Table 1. The following modification to this basic mapping offers advantages when a significant portion of the data values occur near the boundaries of the dynamic range.

For simplicity let $\hat{x}$ denote the prediction of a pixel which takes on the value x. The prediction error is then

$$\Delta = x - \hat{x}. \tag{4}$$

Because the possible values of x and $\hat{x}$ are constrained to the range $[0, x_{MAX}]$, so are the values of Δ. In fact

$$-\hat{x} \leq \Delta \leq x_{MAX} - \hat{x}. \tag{5}$$

Thus the inequalities in (2) are only good approximations out to the limits specified in (5). Outside this range the basic mapping of Δ to δ assigns numbers to events which can't happen, thus violating the desired goals in (1). After some manipulation a modification to (3) which takes advantage of these observations can be derived. The result is given as

$$\delta = \begin{cases} 2\Delta - 1 & \text{if} \quad 0 < \Delta \leq \hat{x} \\ x & \text{if} \quad \Delta > \hat{x} \\ \\ 2|\Delta| & \text{if} \quad \hat{x} - x_{MAX} \leq \Delta \leq 0 \\ x_{MAX} - x & \text{if} \quad \hat{x} - x_{MAX} > \Delta. \end{cases} \tag{6}$$

Image Noiseless Coder Structure

The basic overall image noiseless coder structure is illustrated in Fig. 1 where the variable length coding operations which follow the reversible preprocessing have been denoted collectively as $\psi[\cdot]$.† Note that an initial reference pixel needs to be sent to assure complete reversibility from a sequence of differences.

Practical Performance Measure

Let $\tilde{P}_\Delta$ and $\tilde{P}_\delta$ be the probability distributions for the occurrence of Δ and δ samples in Fig. 1. By the one-to-one mapping in Table 1, $\tilde{P}_\Delta$ and $\tilde{P}_\delta$ are equivalent. Then the entropy of distributions is given by

$$H_\Delta = H_\delta = -\sum_j p_j \log_2 p_j \text{ bits/sample} \tag{7}$$

†That is, $\psi[\cdot]$ denotes the reversible operations that map an input sequence $\tilde{X}$ into coded sequence $\psi[\tilde{X}]$. Subsequent discussion will subscript and superscript ψ to denote different coding algorithms.

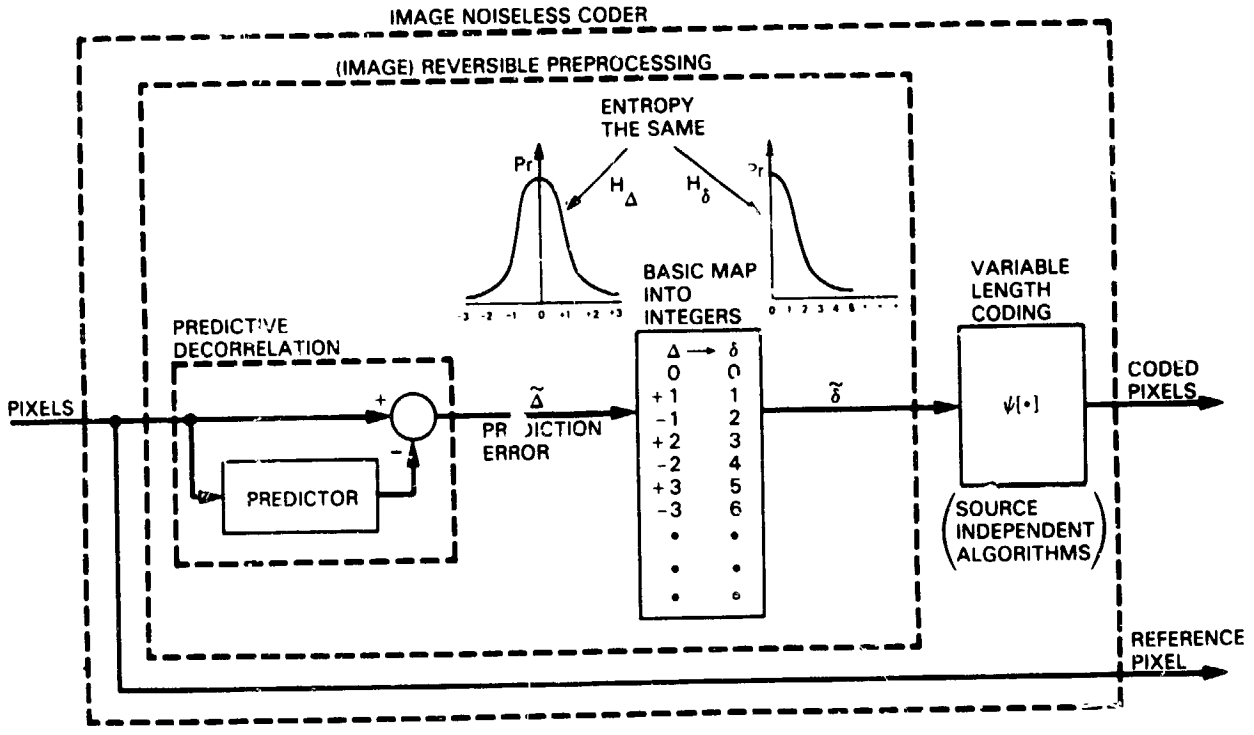

Fig. 1. Basic Image Noiseless Coding Structure.

where the $\{p_j\}$ are the probability values making up $\tilde{P}_\Delta$ and $\tilde{P}_\delta$. Entropies arising from the two predictors in Table 1 are generally called one and two-dimensional differential entropies respectively. Since H_δ is really an average measure of uncertainty in $\tilde{\delta}$ samples, the two-dimensional predictor generally yields a lower entropy.

The entropy, H_δ, provides a practical measure of performance for code operators $\psi[\cdot]$ regardless of how the sequences, δ, originated. However, for the imaging application H_δ (which equals H_Δ) also provides a practical measure of overall "image noiseless coder" performance. Specifically, H_δ represents a bound to the best performance (bits/sample, bits/pixel) of any coding algorithm which treats the δ's as an independent, stationary sequence of samples. It is not an absolute bound to performance unless the latter conditions are true. In the imaging application the independence condition is certainly well approximated by the predictive decorrelation but the obvious fluctuations in scene activity preclude stationarity. This just means that performance under a "measured" H_δ may be possible under some conditions if the coding algorithms, $\psi[\cdot]$, did not rely on stationarity.

Quite obviously, there are both long term and short term variations in image activity so that the value of a measured H_δ depends much on the span of data over which it is measured. In this paper we will deal only with entropies of distributions derived from complete individual images. Performance close to such measured entropies can generally be considered a good guide to efficient performance. The key practical problem in the specification of $\psi[\bullet]$ is to assure that such efficient performance can be expected for all images encountered. In Voyager's case $\psi[\bullet]$ must also meet severe implementation constraints.

Observe from (7) that for imaging applications there is a one-to-one correspondence between performance in bits per δ sample and bits per input pixel. However, $\tilde{\delta}$ sequences with the same characteristics could have originated from sources other than imaging. Hence to emphasize this broader applicability, results will primarily be noted in bits/sample. We will switch to bits/pixel (b/p) when referring to specific imaging results.

Coder Operation at Fixed Line Rates

Another restriction placed on the operation of a Voyager image noiseless coder is that it must operate at pre-established line rates (bits/line). But as already noted, data entropy varies and consequently the number of bits used by an **efficient** noiseless coder will also vary, from line-to-line and image to image. The primary mode to satisfy these format constraints will be to simply truncate a line when its allotted bits have been used up.† Special modes are being considered to more intelligently edit data when an object of interest does not fill a field of view. However, most of the investigations of coding efficiency here will deal with complete images.

COMMUNICATION ERROR EFFECTS

The effect of communication errors on data compressed by the noiseless coding algorithms treated here is significantly more dramatic than on uncompressed imaging. Instead of affecting only a single pixel, an error may confuse a noiseless decompressor so that succeeding pixels may be wrong until the system is reinitialized (e.g., once per

†Ref. 7 describes a rate controlled algorithm developed for the Galileo mission which adaptively changes local quantization to avoid truncation of lines.

line). These effects can be countered to a certain extent by incorporating more frequent updates and using a more sophisticated decompressor that finds errors by looking for the start of anomalous data. However, we will ignore these possibilities here since the primary communication mode at Uranus and Neptune should be "virtually error free". This is provided by a sophisticated concatenated channel incorporating an outer Reed-Solomon block code (symbol size J = 8, error-correction E = 16) coupled with the well known convolutionally coded-Viterbi decoded inner channel. References 8 and 9 provide discussions on the system implications of this channel for which the framework of international standardization has now been established. Numerous papers further treating performance and implementation for space applications are cited in Ref. 8.

II. BASIC ADAPTIVE VARIABLE LENGTH CODING

The remaining problem in the definition of noiseless coders which represent images efficiently is the specification and performance evaluation of the operations denoted by $\psi[\cdot]$ in Fig. 1 which code preprocessed $\tilde{\delta}$ sequences efficiently. This section will motivate the definition of new, computationally improved algorithms by introducing some basic coding operations, definitions and notation originating in Ref. 1. Subsequently, notation will be extended to include both the new and old algorithms under one definition. The performance of several alternatives will be graphically compared using a broad test set of images. Unless noted otherwise, all performance results noted subsequently assume image preprocessing based on the one-dimensional prediction in Table 1 and the standard δ mapping in (3). Results for variations to this approach will be explicitly discussed in Section VI.

SOME NOTATION CONVENTIONS

Concatenation

If $\tilde{X}$ and $\tilde{Y}$ are two sequences of samples then we can form a new sequence $\tilde{Z}$ by running them back to back as

$$\tilde{Z} = \tilde{X} * \tilde{Y} \tag{8}$$

using the asterisk as our basic indication of concatenation. However, the * will be omitted occasionally when no confusion should result.

Length of a sequence. Any sequence of non-binary samples can be represented by a sequence of bits using the familiar binary representations which use a fixed number of bits. Without any anticipated confusion, the function $\mathscr{L}(\cdot)$ will be used to specify the length of any sequence in samples or bits (of its standard binary representation) as required. For example, if $\tilde{X}$ is a sequence of J samples requiring m bits/sample to represent we may take

$$\mathscr{L}(\tilde{X}) \tag{9}$$

as J samples or mJ bits. Of course there is only one possibility if $\tilde{X}$ is already a binary sequence.

FUNDAMENTAL SEQUENCE

Consider first an extremely simple variable length coding operation which will become central in later developments. Define the code word function fs[•] by

$$\text{fs}[i] \equiv \overbrace{0\,0\,0\,\ldots\,0\,0\,0}^{\text{i zeroes}}\,1 \tag{10}$$

where $i \geq 0$ is an input integer. The length of "codeword" fs[i] is

$$\ell_i = \mathscr{L}(\text{fs}[i]) = i+1 \text{ bits.} \tag{11}$$

Let

$$\tilde{X} = x_1\, x_2 \ldots x_J \tag{12}$$

denote a sequence of samples meeting the conditions of preprocessing described earlier. Then, the coding of $\tilde{X}$ using fs[•] on each sample yields the fundamental sequence of $\tilde{X}$

$$\psi_1[\tilde{X}] = \text{FS}[\tilde{X}] \equiv \text{fs}[x_1] * \text{fs}[x_2] * \ldots * \text{fs}[x_J]. \tag{13}$$

That is, $\psi_1[\bullet]$ or FS[•] denote the operations of applying (10) to each symbol of a sequence. The length of a fundamental sequence is

$$F_0 = \mathscr{L}(\text{FS}[\tilde{X}]) = J + \sum_{j=1}^{J} x_j. \tag{14}$$

Observations

Because of the assumed probability ordering of input symbols in (1) and the codeword lengths in (11), shorter codewords will be used more often than longer ones. Note that the definition in (10) does not depend on alphabet size.

FS Performance

A plot of the typical average **per sample** performance of code operator $\psi_1[\cdot] = FS[\cdot]$ is shown in Fig. 2 as a function of measured data entropy H_δ. The graph was derived from the results of preprocessing many forms of data such that condition (1) was well approximated.

Note that performance is efficient (close to the entropy) in the range of roughly 1.5 to 3.0 bits/sample, but rapidly becomes inefficient outside this range. Unfortunately, Voyager Uranus and Neptune encounters may generate some images with entropies exceeding 4 b/p although the majority are expected to be at much lower values. Other applications can have much broader swings in data entropy. Thus in general code operator $\psi_1[\cdot] = FS[\cdot]$ is inadequate by itself. Additional code options are needed to extend the range of efficient performance.

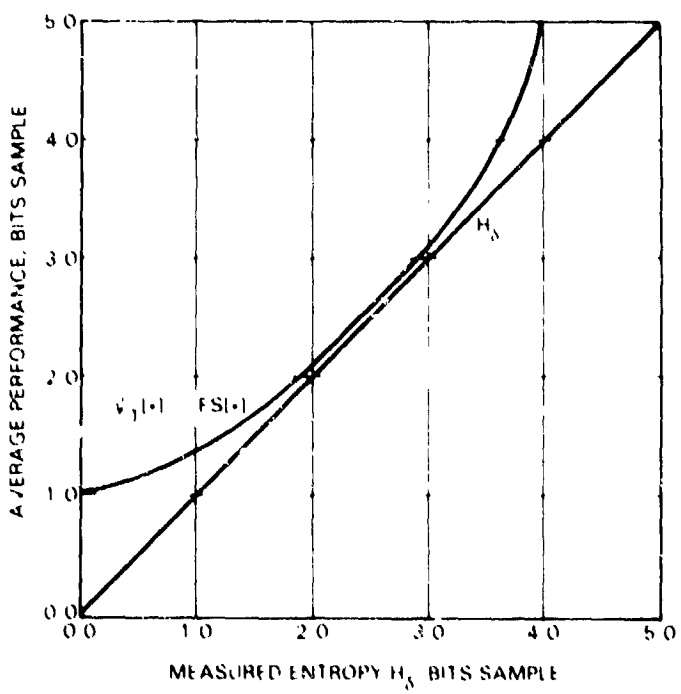

Fig. 2. Average FS Performance.

UNITY CODE OPERATOR $\psi_3[\cdot]$

A trivial code option is obtained by defining

$$\psi_3[\tilde{X}] \tag{15}$$

as any fixed length binary representation of $\tilde{X}$. In the simplest case we can take $\psi_3[\tilde{X}]$ as $\tilde{X}$ itself so that

$$\psi_3[\tilde{X}] = \tilde{X}. \tag{16}$$

However, in some applications such as imaging it may be advantageous to take $\psi_3[\tilde{X}]$ as a fixed length binary representation **before** reversible preprocessing. $\psi_3[\cdot]$ can be interpreted as a "backup" code operation to be used when all else fails. Obviously, the performance of $\psi_3[\cdot]$ is independent of entropy.

OPERATORS $\psi_0[\cdot]$ AND $\psi_2[\cdot]$

Code operators $\psi_0[\cdot]$ and $\psi_2[\cdot]$ (also known as "code fs bar" and "code fs") offer efficient performance in entropy ranges below 1.5 bits/sample and above 3.0 bits/sample respectively as illustrated in Fig. 3. They are defined in Ref. 1.

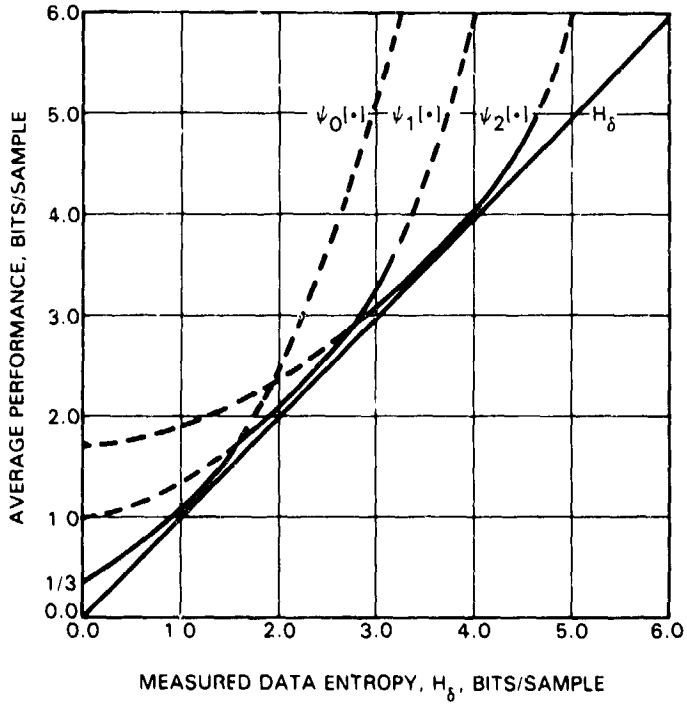

Fig. 3. Average Performance, $\psi_0[\cdot]$, $\psi_1[\cdot]$ and $\psi_2[\cdot]$.

THE BASIC COMPRESSOR, $\psi_4[\cdot]$

An adaptive coder called the Basic Compressor[1]-[3] can be defined by choosing between these four options as

$$\psi_4[\tilde{X}] = BC[\tilde{X}] = ID * \psi_{ID}[\tilde{X}] \tag{17}$$

where the concatenated ID is assumed to be a 2-bit binary number whereas, as a subscript to ψ it takes on the values 0, 1, 2 or 3.

The most straightforward, and in fact optimum, selection procedure is to simply choose the code operator output sequence which is shortest. Other simplified procedures discussed in Ref. 1 provide nearly identical results.

Performance

The average performance of the Basic Compressor, $\psi_4[\cdot]$, used on a broad set of 8 b/p Voyager images (from Saturn encounter) and other selected data is shown in Fig. 4 where the Voyager data points are indicated by the symbol x. One-dimensional prediction (Table 1) and the standard mapping (Table 2) is assumed. The impact of two-dimensional prediction and the modified mapping in (6) will be noted later in Section VI. The portion of the image test set derived from Voyager images is shown in Fig. 5. Subsequent performance graphs using this same test set will omit the data points although the anticipated Voyager entropy range will continue to be specified.

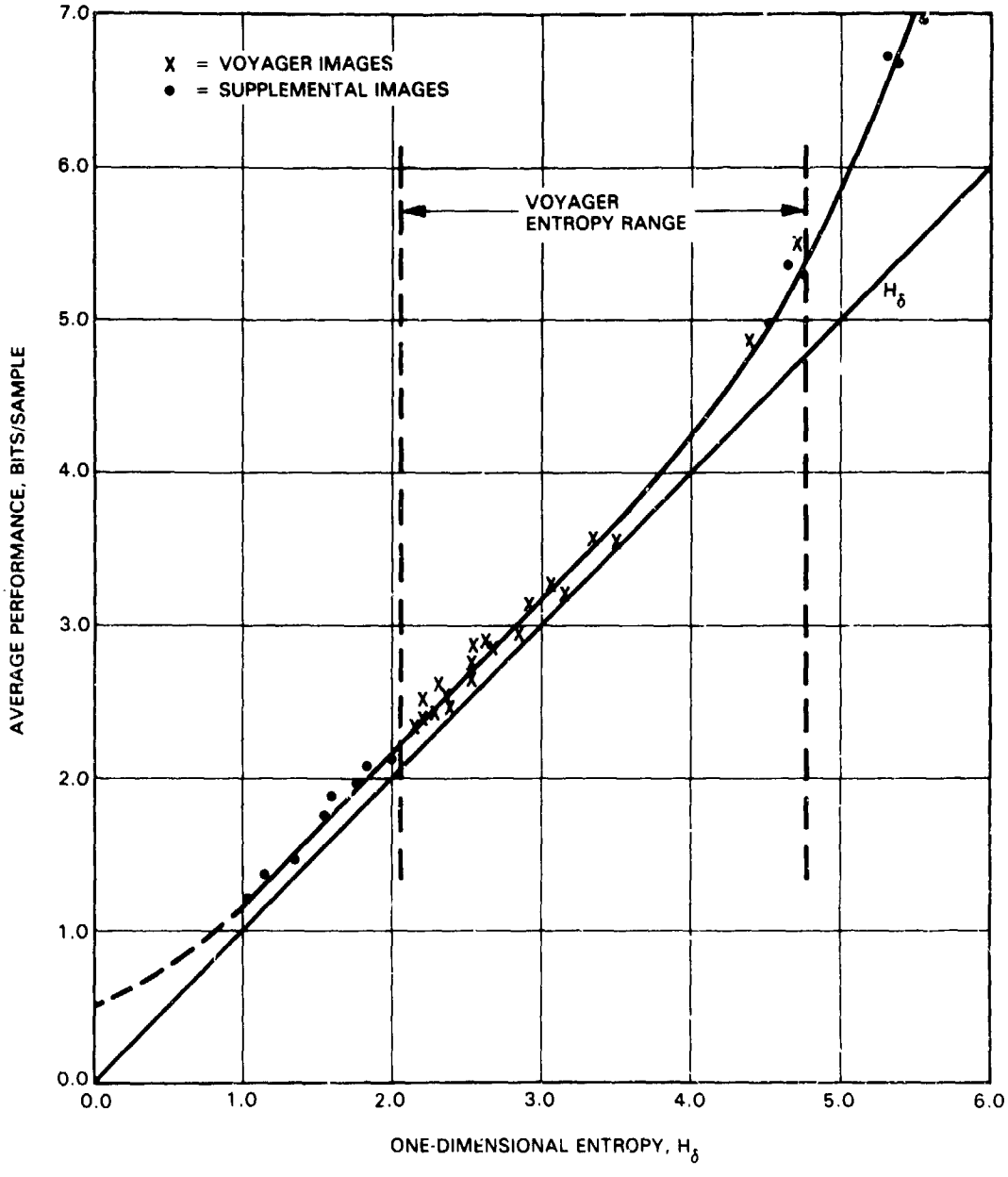

Fig. 4. Average Performance, $\psi_4[\cdot]$.

Fig. 5. Voyager Test Set.

III. CODE STRUCTURE FOR HIGH ENTROPIES

$\psi_2[\bullet]$ gives the Basic Compressor efficient performance out to about 4 b/p, where the performance curve begins to move away from the entropy line. For Voyager, this covers the entropy range expected for most images. However, in some cases, one-dimensional image entropies might be as high as 5 b/p and, at a more local line-by-line level, even higher.

References 1-3 define additional code operators, $\psi_8[\bullet]$ and $\psi_8^J[\bullet]$, which combine "split-sample" modes with $\psi_4[\bullet]$ or $\psi_2[\bullet]$ alone to extend efficient performance to any higher entropy range. However, difficulties in implementing $\psi_2[\bullet]$ under Voyager flight data system computation constraints led us to investigate combinations of the split-sample modes with the simpler fundamental sequence operator, $\psi_1[\bullet]$. To this end we will first develop an appropriate code structure and then investigate performance implications of various options. Previously defined algorithms in Refs. 1-3 will fit within this code structure as will the extremely simple "FAST Compressor" destined for the Voyager Uranus encounter. Later sections will further extend these split-sample mode definitions.

BASIC SPLIT-SAMPLE OPERATIONS

Following Ref. 1, let $\tilde{M}_0^n$ be a sequence of N samples, represented by n bits/sample using a standard binary representation (a sign bit, if present, is assumed to be the most significant bit). Define the basic split-sample operator $SS_0^{n,k}[\bullet]$ by

$$SS_0^{n,k}[\tilde{M}_0^n] = \left\{\tilde{L}_k^0,\ \tilde{M}_0^{n,k}\right\} \tag{18}$$

where

$$\tilde{L}_k^0 = S_{A,0}^{n,k}[\tilde{M}_0^n] = \left\{\begin{array}{l}\text{N sample}\\ \text{sequence made}\\ \text{up of the k}\\ \text{least significant}\\ \text{bits of each}\\ \tilde{M}_0^n \text{ sample}\end{array}\right. \tag{19}$$

and $\tilde{M}_0^{n,k}$ is simply all the remaining most significant bits of each sample after removing the first k. By this definition the parameter n is really not crucial. However, its inclusion is useful in keeping track of more complex operations later. Thus we define

$$\tilde{M}_0^{n,k} = S_{B,0}^{n,k}[\tilde{M}_0^n] = \begin{cases} \text{N sample sequence} \\ \text{made up of the} \\ \text{n-k } \textbf{most} \text{ significant} \\ \text{bits of each} \\ M_0^n \text{ sample} \end{cases} \tag{20}$$

and where $SS_0^{n,0}[\tilde{M}_0^n] = \tilde{M}_0^n$. These operations are illustrated in Fig. 6.†

Alternative Structures for Imaging

For imaging applications the split-sample operations can be used for coding purposes using the two different arrangements in Fig. 7.

The structure in the upper portion (Structure A) assumes that the predictive preprocessing described in earlier sections **precedes** the split-sample operator $SS_0^{n,k}[\cdot]$, forming $\tilde{M}_0^n$ (equivalent to the sequence δ in Fig. 1). Operator $SS_0^{n,k}[\cdot]$ then generates sequences $\tilde{L}_k^0$ and $\tilde{M}_0^{n,k}$ as defined in (19) and (20). The latter sequences are passed on for further coding to be discussed momentarily.

---

†If m is an individual n bit sample of $\tilde{M}_0^n$ we can write

$$m = b_{n-1} \cdot 2^{n-1} + \ldots + b_k \cdot 2^k + b_{k-1} \cdot 2^{k-1} + \ldots + b_0 \cdot 2^0$$

$$= 2^k \left(\sum_{j=0}^{n-k-1} b_{k+j} \cdot 2^j \right) + \sum_{j=0}^{k-1} b_j \cdot 2^j$$

$$= 2^k \phi_M + \phi_L$$

where ϕ_M and ϕ_L are elements of $\tilde{M}_0^{n,k}$ and $\tilde{L}_k^0$ respectively.

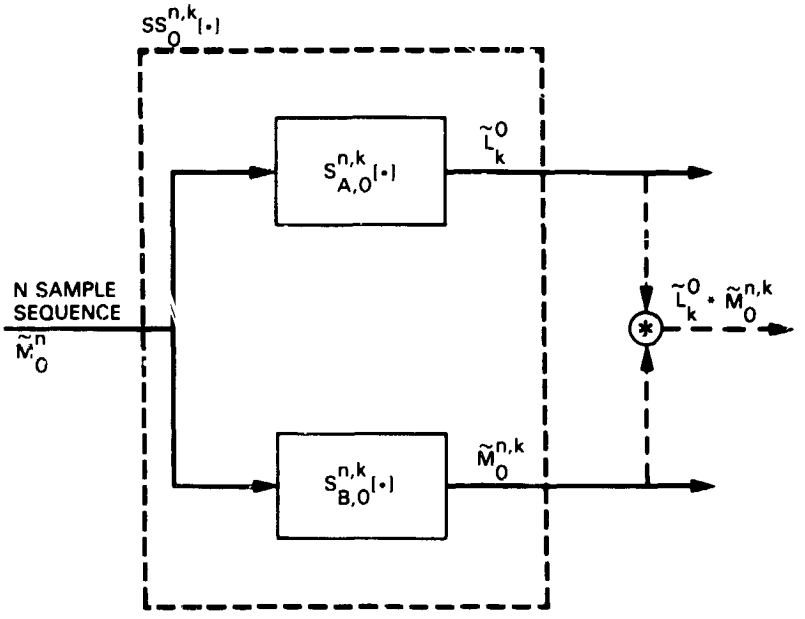

Fig. 6. Basic Split-Sample Operator, $SS_0^{n,k}[\cdot]$.

By contrast the arrangement in the lower part of the diagram (Structure B) assumes that the split of least and most significant bits occurs first, followed by image reversible preprocessing. Using script letters to accentuate the difference, we let $\tilde{\mathscr{M}}_0^n$ be the original n b/p image data which is applied directly to $SS_0^{n,k}[\cdot]$. Following the same definitions in (19) and (20), we take the output of $SS_0^{n,k}[\cdot]$ as $\tilde{\mathscr{L}}_k^0$ and $\tilde{\mathscr{M}}_0^{n,k}$ respectively. We then define

$$\tilde{\approx}_k^0 \equiv \tilde{\mathscr{L}}_k^0 \tag{21}$$

and

$$\tilde{\approx}_0^{n,k} \tag{22}$$

as the result of image reversible preprocessing operations on $\tilde{\mathscr{M}}_0^{n,k}$.

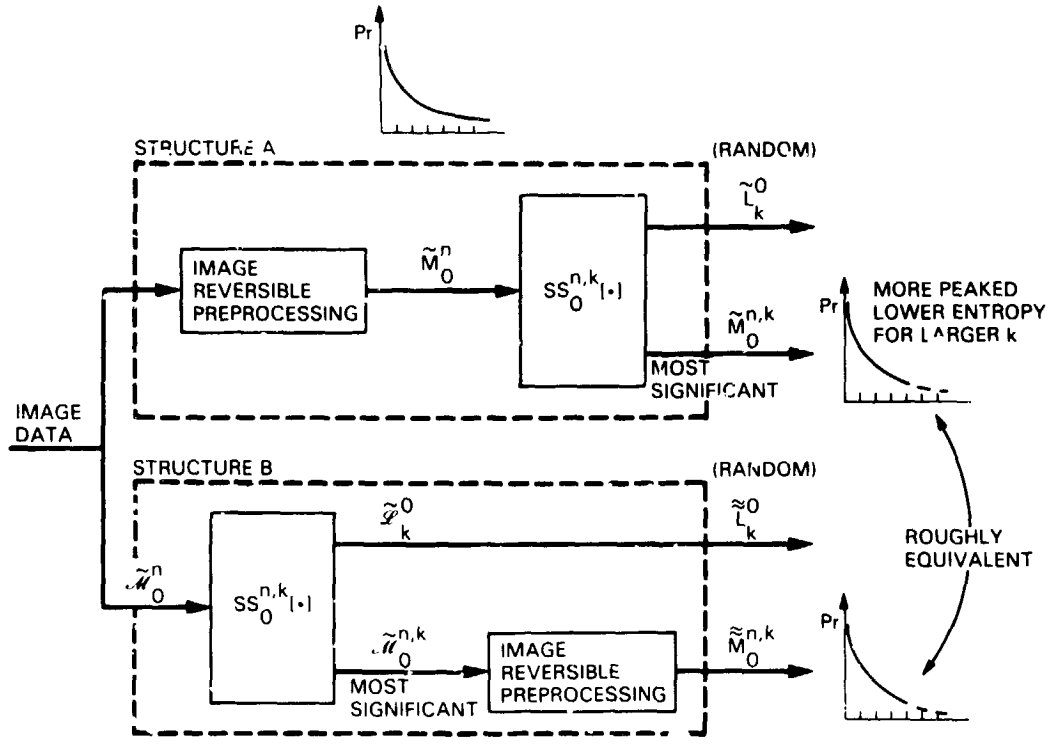

Fig. 7. Alternative Split-Sample Structure for Image Data.

Structure A or Structure B. The statistical characteristics of $\tilde{L}_k^0$ and $\tilde{M}_0^{n,k}$ are **roughly equivalent** to those of $\tilde{\tilde{L}}_k^0$ and $\tilde{\tilde{M}}_0^{n,k}$. Thus, for **image data sources**, any coding algorithms specified using $\tilde{L}_k^0$ and $\tilde{M}_0^{n,k}$ will yield very nearly the same average performance if $\tilde{\tilde{L}}_k^0$ and $\tilde{\tilde{M}}_0^{n,k}$ are substituted. Specifically, numerous comparisons indicate that, while close, Structure A can offer an advantage of as much as 0.1 b/p over Structure B (the largest advantages occurring at entropies above 4 b/p). This difference may be small enough in some applications for an implementer to choose one approach over the other based solely on hardware and software considerations. However, keeping this tradeoff in mind, we will henceforth assume Structure A when quoting results for image data. Structure A is in fact more appealing since it maintains the original arrangement established in Fig. 1 where image preprocessing functions are completely separated from variable length coding functions. We again need only define and evaluate algorithms for coding preprocessed $\tilde{M}_0^n$ sequences (δ in Fig. 1). Results are not dependent on the source of $\tilde{M}_0^n$ sequences and are thus broadly applicable to non-imaging sources which can be appropriately preprocessed.

Preprocessing Assumption. Until Section VI, all stated results will additionally assume that the image preprocessing which generates $\tilde{M}_0^n$ sequences is limited to the one-dimensional prediction of Table 1 and the standard mapping of error signals into integers given by (3).

Motivation for Split-Sample Coding

Now focussing on Structure A, since $\tilde{M}_0^n$ sequences are the result of appropriate image preprocessing, sample distributions will exhibit the desired characteristics specified by (1). But by assumption the entropy of $\tilde{M}_0^n$ sequences is too high to be coded efficiently using algorithms specified in earlier sections. However, as k is increased, the $\tilde{M}_0^{n,k}$ sequences continue to **retain** the desired ordering at **decreasing** entropy values (more peaked distributions). At some value of k the entropy of $\tilde{M}_0^{n,k}$ sequences will drop low enough to lie within the efficient operating range of operators $\psi_0[\cdot]$, $\psi_1[\cdot]$ or $\psi_2[\cdot]$, suggesting an efficient means for coding that portion of the data.

An appropriate means for coding the remaining $\tilde{L}_k^0$ sequences is trivial. Until the decreasing entropy of $\tilde{M}_0^{n,k}$ sequences reaches about 3 bits/sample, the least significant bits appear totally random and are therefore already optimally coded. Consolidating these observations leads to a set of split-sample code operators, $\psi_{i,k}[\cdot]$.

OPERATOR $\psi_{i,k}[\cdot]$

In the following discussions we will modify the original definition of split-sample coding to allow for additional generality. Split-sample code operator $\psi_{i,k}[\cdot]$ is defined by†

$$\psi_{i,k}[\tilde{M}_0^n] = \tilde{L}_k^0 * \psi_i[\tilde{M}_0^{n,k}]. \tag{23}$$

Equation (23) means that coding N sample sequence $\tilde{M}_0^n$ by $\psi_{i,k}[\cdot]$ is accomplished by using $\psi_i[\cdot]$ to code the (n-k) most significant bit samples and prefixing the result with a sequence of all the k least significant bits.‡ The coding structure for $\psi_{i,k}[\cdot]$ is given in Fig. 8.

---

†Observe that $\psi_{i,k}[\cdot]$ will later be generalized to $\psi_{i,k,k'}[\cdot]$ with equivalence when $k' = n-k$. We will continue with the simpler notation for now.

‡$\tilde{L}_k^0$ can be placed before or after $\psi_i[\tilde{M}_0^{n,k}]$.

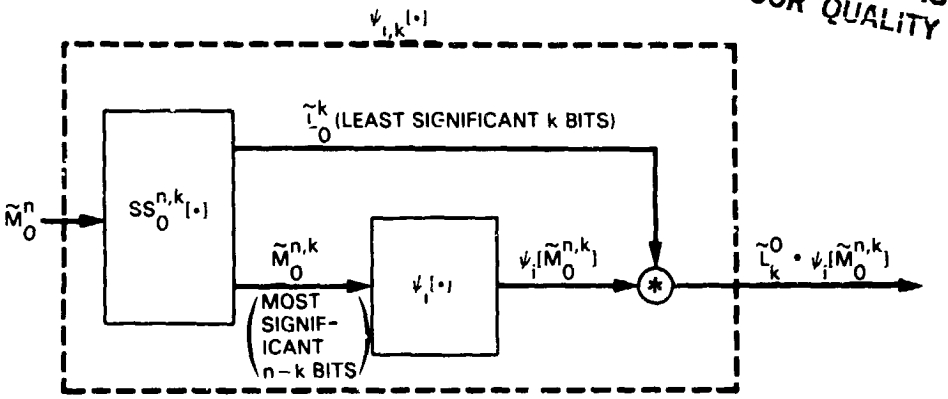

Fig. 8. Split-Sample Coder, $\psi_{i,k}[\cdot]$.

Special Cases[10]

A case of special interest to the Voyager mission is where we take $\psi_i[\cdot]$ in (23) to be non-existent. That is, we take i to be ϕ, indicating the empty set. Equation 23 becomes

$$\psi_{\phi,k}[\tilde{M}_0^n] = \tilde{L}_k^0. \tag{24}$$

This will code $\tilde{M}_0^n$ **noiselessly** with k bits/sample **if** it is a priori known that the most significant n-k bits of each $\tilde{M}_0^n$ sample never change (i.e., remain equal to zero). Observe that (24) further reduces to already defined unity code operator $\psi_3[\cdot]$ if we let k = n, that is

$$\psi_{\phi,n}[\tilde{M}_0^n] = \tilde{L}_n^0 = \psi_3[\tilde{M}_0^n]. \tag{25}$$

We will later define an adaptive coder, called the "Fast Compressor," based on (24).

Before continuing, it is worthwhile noting some other equivalences which occur under special conditions. If we take k = 0 in (23) we get

$$\begin{aligned} \psi_{0,0}[\cdot] &\equiv \psi_0[\cdot] \\ \psi_{1,0}[\cdot] &\equiv \psi_1[\cdot] \\ \psi_{2,0}[\cdot] &\equiv \psi_2[\cdot] \\ \psi_{\phi,n}[\cdot] = \psi_{3,0}[\cdot] &\equiv \psi_3[\cdot] \\ \psi_{4,0}[\cdot] &\equiv \psi_4[\cdot] \\ \psi_{5,0}[\cdot] &\equiv \psi_5[\cdot] \end{aligned} \qquad (26)$$

(see Ref. 1).

The reader familiar with Ref. 1 will also note that the definition of $\psi_{i,k}[\cdot]$ is essentially the same as $\psi_7^m[\cdot] \equiv \psi_7^{n-k}[\cdot]$ defined in Refs. 1-3. This new form allows more flexibility in new definitions to follow.

OPERATOR $\psi_{11}[\cdot]$

We now define an adaptive operator which chooses between the split-sample modes $\psi_{i,k}[\cdot]$ just defined, or any other previously defined operators. $\psi_{11}[\cdot]$ coding of sequences $\tilde{M}_0^n$ takes the form

$$\psi_{11}[\tilde{M}_0^n] \equiv ID * \psi_{ID}[\tilde{M}_0^n] \qquad (27)$$

where the function of 'ID' is to identify which (previously defined) code operator is to be used to represent $\tilde{M}_0^n$.

$\psi_{11}[\cdot]$ takes on exactly the same form as the "Basic Compressor" $\psi_4[\cdot]$ and is in fact equivalent if 'ID' is restricted to the options 0, 1, 2 and 3. But the assumption in (27) is that 'ID' can identify any previously defined code operator, including $\psi_{i,k}[\cdot]$ in (23) or (24). As a binary prefix to $\psi_{ID}[\tilde{M}_0^n]$ in (27) 'ID' can be represented by a fixed length binary code.

IV. PERFORMANCE FOR SELECTED $\psi_{11}[\cdot]$

In this section we will investigate the performance impact of different sets of code options for adaptive code operator, $\psi_{11}[\cdot]$. All performance graphs include appropriate bit/sample cost for code option identifiers.

OPTIONS USING $\psi_0[\cdot] - \psi_3[\cdot]$

As already noted, choosing options $\psi_0[\cdot]$, $\psi_1[\cdot]$, $\psi_2[\cdot]$ and $\psi_3[\cdot]$ (i.e., ID = 0, 1, 2 or 3) makes $\psi_{11}[\cdot]$ equivalent to the Basic Compressor $\psi_4[\cdot]$. The performance of $\psi_4[\cdot]$ was illustrated in Fig. 4.

OPTIONS USING $\psi_{4,k}[\cdot]$ AND $\psi_{5,k}[\cdot]$

By expanding the allowed code options to include split-sample modes $\psi_{4,0}[\cdot]$, $\psi_{4,1}[\cdot]$, $\psi_{4,2}[\cdot]$, etc. (or $\psi_{5,0}[\cdot]$, $\psi_{5,1}[\cdot]$, $\psi_{5,2}[\cdot]$, ..., see Ref. 1)† operator $\psi_{11}[\cdot]$ becomes equivalent to $\psi_8[\cdot]$ originally defined in Refs. 1-3. The performance of such a $\psi_{11}[\cdot]$ with eight options is shown in Fig. 9 using the same test set described earlier (photographs of the Voyager portion of this set were displayed in Fig. 5).

Note that allowing options $\psi_{4,0}[\cdot]$, $\psi_{2,1}[\cdot]$, $\psi_{2,2}[\cdot]$, ... yields yet another equivalence to $\psi_8^J[\cdot]$ (with J = N) defined in Refs. 1-3. $\psi_8^J[\cdot]$ performance is roughly equivalent to $\psi_8[\cdot]$ but is somewhat simpler.

OPTIONS USING ONLY $\psi_{1,k}[\cdot]$ AND $\psi_3[\cdot]$

Figure 9 illustrates that the originally defined $\psi_8[\cdot]$ operators assure efficient performance over any entropies exceeding about 0.7 bits/sample. But the $\psi_8[\cdot]$ operators rely primarily on $\psi_2[\cdot]$ to code the split most-significant bit sequences $\tilde{M}_0^{n,k}$ either as part of the Basic Compressor or directly in the case of $\psi_3^J[\cdot]$. We now investigate the sole use of the simpler fundamental sequence operator $\psi_1[\cdot]$ for that purpose.

Assuming the desire for a fixed length binary representation for the 'ID' prefix in (27) we have two practical possibilities to investigate. A three-bit 'ID' allows the identification of up to eight options whereas a two-bit 'ID' allows up to four. Consider the former case first.

---

†$\psi_5[\cdot]$ partitions a sequence into smaller blocks which are separately coded using $\psi_4[\cdot]$.

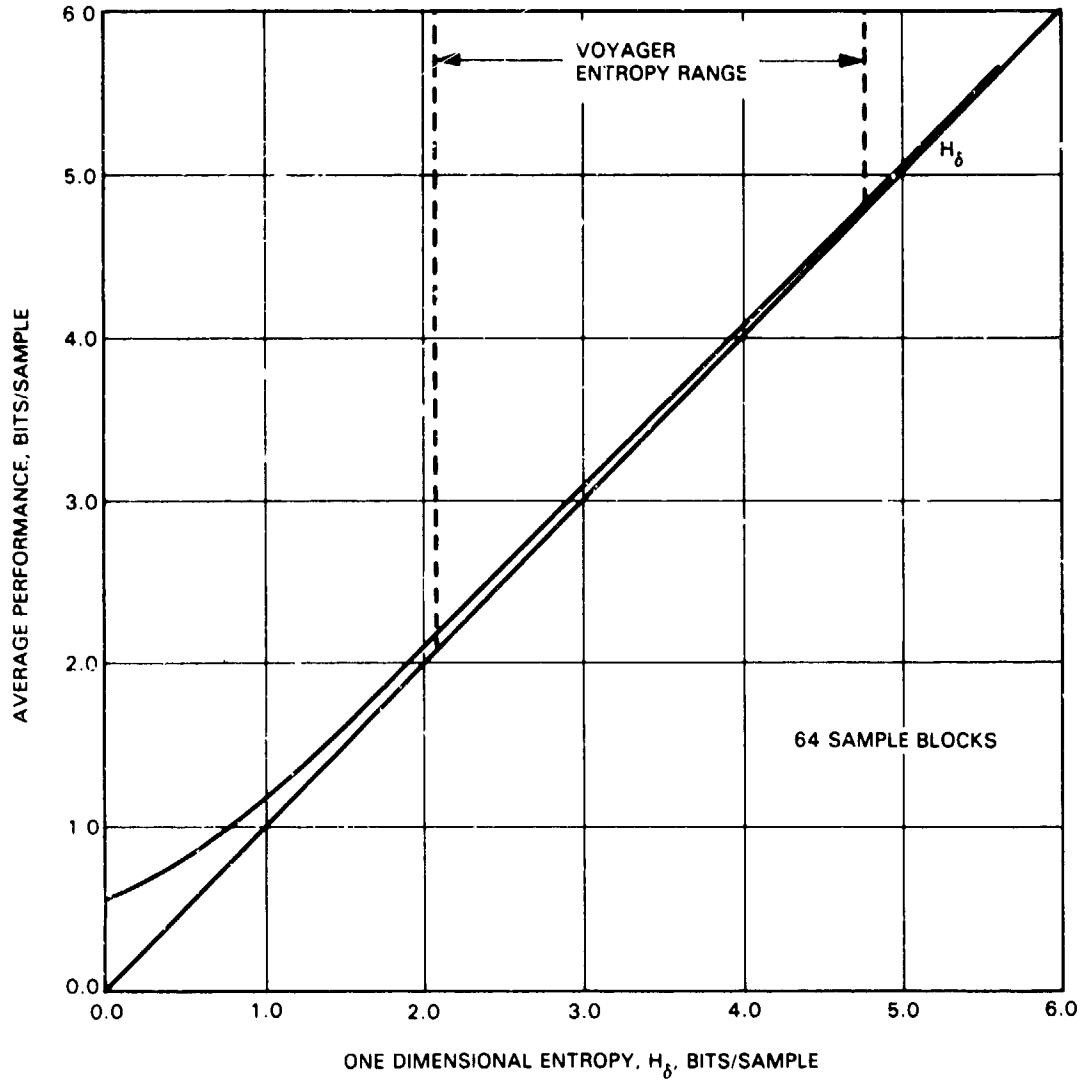

Fig. 9. Average Performance, $\psi_{11}[\cdot] = \psi_8[\cdot]$ with Eight Options: $\psi_{5,k}[\cdot]$, k = 0, 1, 2, . . . 6, $\psi_3[\cdot]$.

Eight-Option Codes

We include the following code operator options for a $\psi_{11}[\cdot]$ having 8 options:

$$\psi_0[\cdot],\ \psi_1[\cdot],\ \psi_{1,1}[\cdot],\ \psi_{1,2}[\cdot],$$
$$\psi_{1,3}[\cdot],\ \psi_{1,4}[\cdot],\ \psi_{1,5}[\cdot],\ \psi_3[\cdot]. \tag{28}$$

In actual coding we let the prefix ID in (27) take on the binary values 000, 001, 010, ..., 111 to correspond to the use of operators $\psi_0[\cdot]$, $\psi_1[\cdot]$, $\psi_{1,1}[\cdot]$,..., $\psi_3[\cdot]$ respectively. For example, if $\psi_{1,3}[\cdot]$ was chosen as the most efficient operator to code a sequence $\tilde{M}_0^n$, the output for $\psi_{11}[\cdot]$ would take the form

$$\psi_{11}[\tilde{M}_0^n] = 101 * \psi_{1,3}[\tilde{M}_0^n] \tag{29}$$

which from (23) breaks down further to

$$\psi_{11}[\tilde{M}_0^n] = 101 * \tilde{L}_3^0 * \psi_1[\tilde{M}_0^{n,3}]. \tag{30}$$

The average performance for this selection of code options is shown in Fig. 10. Observe that the inclusion of $\psi_0[\cdot]$ would have no impact on expected Voyager performance and could be deleted for that application. In general, the resulting graph, shown dashed, would intercept the ordinate slightly above 1 bit/sample. The replacement of $\psi_0[\cdot]$ with a very high entropy split-sample mode $\psi_{1,6}[\cdot]$ would have negligible impact on the upper end of this graph for the test set used.

Note that the performance graph for this fundamental sequence based $\psi_{11}[\cdot]$ is nearly identical to that for a $\psi_8[\cdot]$ version which relied on the availability of $\psi_2[\cdot]$ to code the split (most significant) samples. The only difference is a slight advantage at entropies approaching 6 bits/sample, certainly of no consequence to Voyager.

Four-Option Codes

With only four options to choose from it becomes more crucial to pick those options to cover the entropy range of interest. For the Voyager and other similar imaging problems $\psi_0[\cdot]$ can be omitted in favor of a higher entropy option. We consider the latter case first where we choose options

$$\psi_1[\cdot] = \psi_{1,0}[\cdot], \psi_{1,1}[\cdot], \psi_{1,2}[\cdot], \psi_3[\cdot] \tag{31}$$

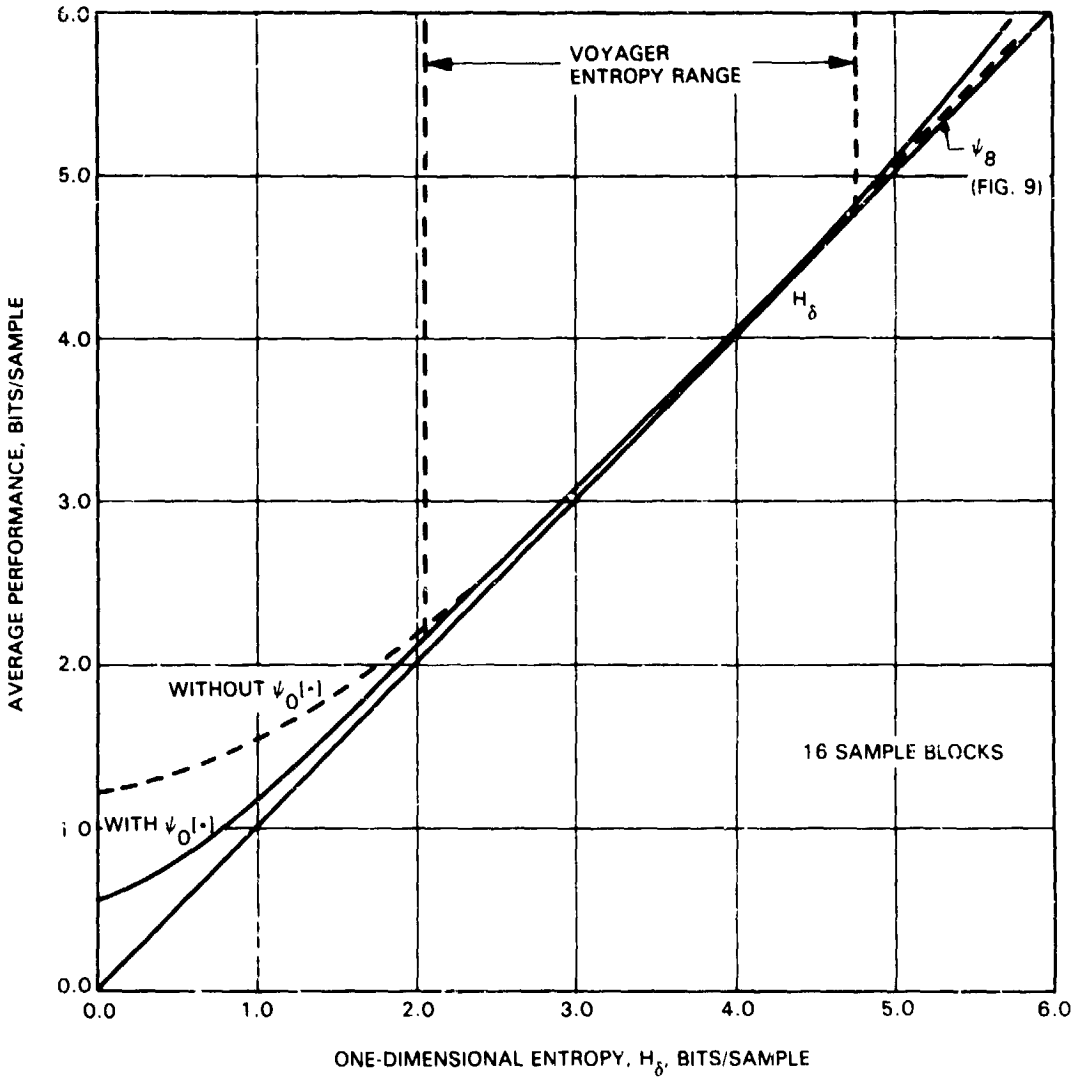

Fig. 10. $\psi_{11}[\cdot]$ Performance with Eight Options: $\psi_0[\cdot]$, $\psi_1[\cdot]$, $\psi_{1,1}[\cdot]$, . . ., $\psi_{1,5}[\cdot]$ and $\psi_3[\cdot]$.

which use only $\psi_1[\cdot]$ as the basic variable length coder. As a means of comparison we also investigate the four options

$$\psi_1[\cdot],\ \psi_2[\cdot],\ \psi_{2,1}[\cdot],\ \psi_3[\cdot] \tag{32}$$

which still includes the use of $\psi_2[\cdot]$. A performance comparison is shown in Fig. 11.

Figure 11 shows that the two sets of options in (31) and (32) yield very similar performance at the lower entropies. However, a growing but small advantage is indicated for the second set incorporating $\psi_2[\cdot]$ at entropies exceeding 4 bits/sample. The performance differential is about 0.3 bits/sample at 5.0 bits/sample entropy.

Now exchange the high entropy options in (31) and (32) for operator $\psi_0[\cdot]$. The resulting code option sets are then

$$\psi_0[\cdot],\ \psi_1[\cdot],\ \psi_{1,1}[\cdot],\ \psi_3[\cdot] \tag{33}$$

and

$$\psi_0[\cdot],\ \psi_1[\cdot],\ \psi_2[\cdot],\ \psi_3[\cdot]. \tag{34}$$

We again recognize the second set as the options making up the Basic Compressor, $\psi_4[\cdot]$. Thus we are now investigating whether a $\psi_{1\,1}[\cdot]$, choosing between the options in (33), can provide a simpler alternative to the Basic Compressor for the range of entropies from 0.7 to 4 bits/sample. A comparison of performance is shown in Fig. 12 where the Basic Compressor results are the same as in Fig. 4.

Figure 12 again illustrates a slight advantage at higher entropies for a four option $\psi_{1\,1}[\cdot]$ which includes $\psi_2[\cdot]$ (in this case the Basic Compressor). The graphs are almost replicas of those in Fig. 11 but shifted downward by 1 bit/sample. But note that Figs. 9 and 10 illustrate that this advantage of incorporating $\psi_2[\cdot]$ can be made negligible over most practical entropy ranges (e.g., Voyager) by incorporating an increased number of split-sample options.

DECISION CRITERIA

The optimum decision criteria for any $\psi_{1\,1}[\cdot]$ configuration is to choose the option which yields the shortest coded sequence. That is

Choose ID such that

$$\mathscr{L}(\psi_{ID}[\tilde{M}_0^n]) = \min_i \mathscr{L}(\psi_i[\tilde{M}_0^n]) \tag{35}$$

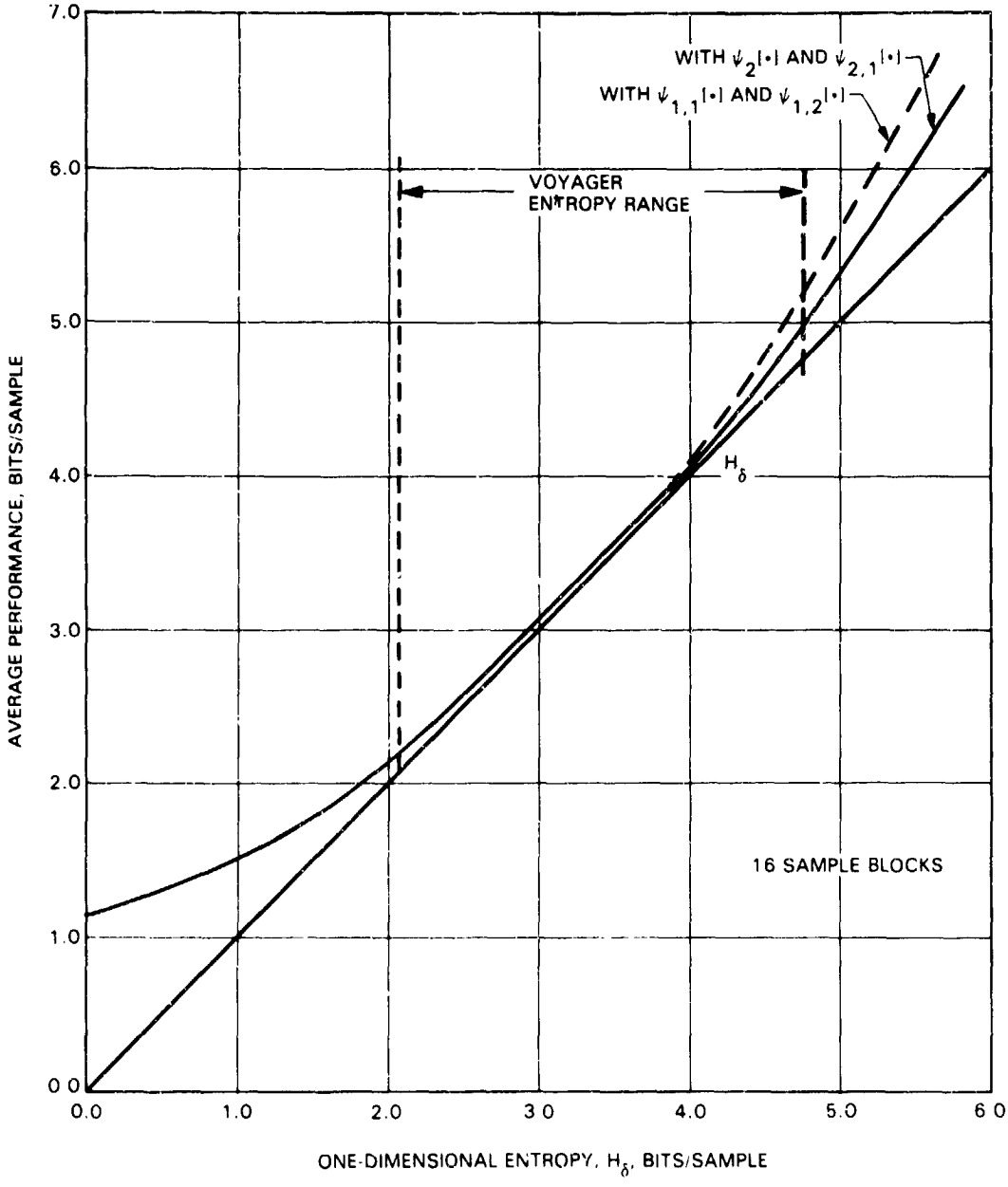

Fig. 11. $\psi_{11}[\cdot]$ Performance with Four Options: $\psi_1[\cdot]$, $\psi_3[\cdot]$ and $\psi_{1,1}[\cdot]$, $\psi_{1,2}[\cdot]$ **OR** $\psi_2[\cdot]$, $\psi_{2,1}[\cdot]$.

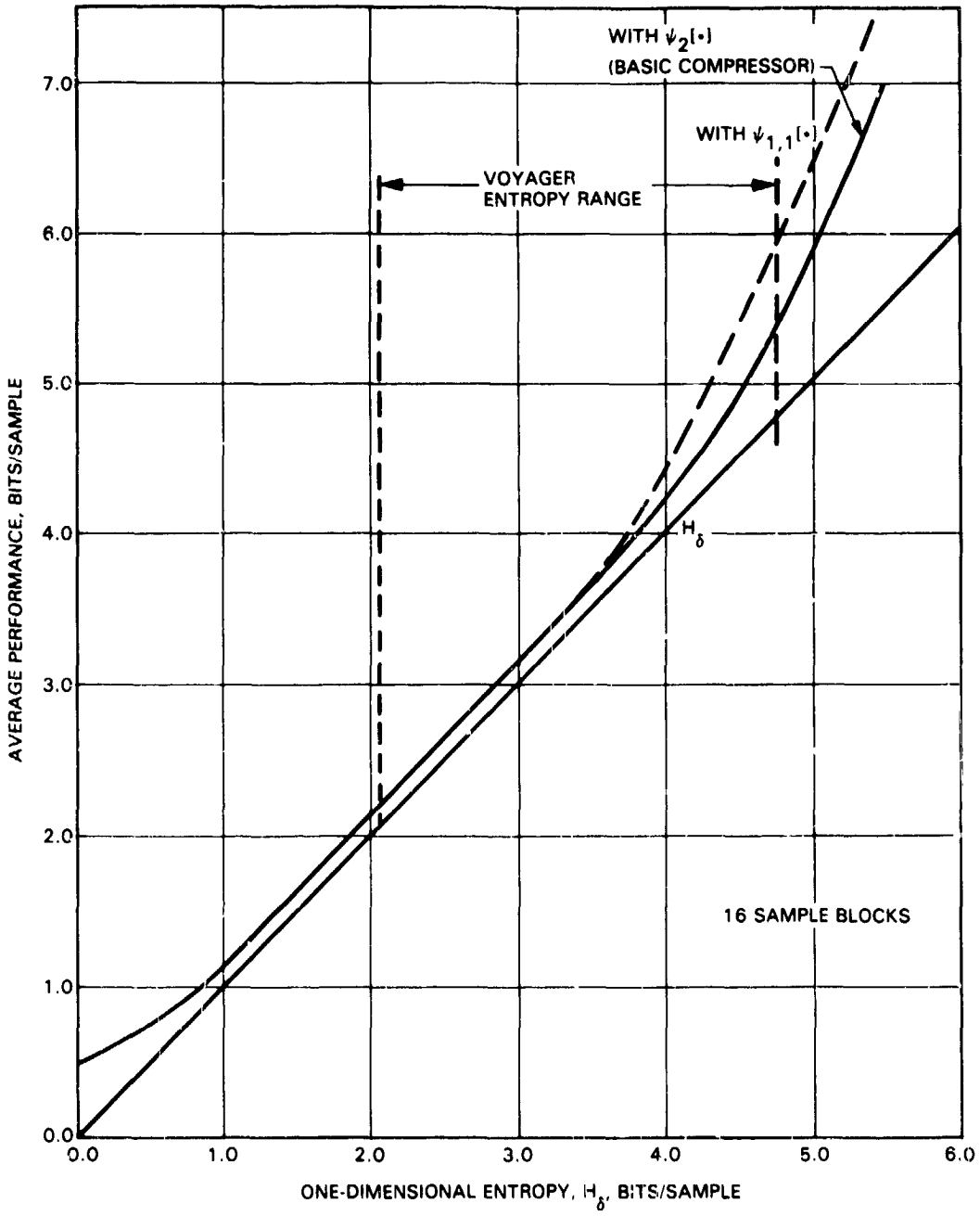

Fig. 12. $\psi_{11}[\cdot]$ Performance with Four Options: $\psi_0[\cdot]$, $\psi_1[\cdot]$, $\psi_3[\cdot]$ and $\psi_{1,1}[\cdot]$ **OR** $\psi_2[\cdot]$.

where i is taken over the set of a priori selected options. A simplified decision rule can sometimes reduce computation requirements by replacing $\mathscr{L}(\psi_i[\tilde{M}_0^n])$ by a more easily obtained estimate $\gamma_i(\tilde{M}_0^n)$. Taking

$$\gamma_i(\tilde{M}_0^n) \approx \mathscr{L}(\psi_i[\tilde{M}_0^n]) \tag{36}$$

the decision rule in (35) is replaced by

Choose ID such that

$$\gamma_{ID}[\tilde{M}_0^n) = \min_i \gamma_i(\tilde{M}_0^n). \tag{37}$$

Such estimates and decision criteria were developed for all code operators defined in Refs. 1-3. In this section we will investigate simplified estimates for split-sample modes $\psi_{1,k}[\cdot]$.

Expanding $\mathscr{L}(\psi_{1,k}[\tilde{M}_0^n])$

By (23), the number of bits required by split-sample mode $\psi_{1,k}[\cdot]$ is given by

$$\mathscr{L}(\psi_{1,k}[\tilde{M}_0^n]) = \mathscr{L}(\tilde{L}_k^0) + \mathscr{L}(\psi_1[\tilde{M}_0^{n,k}]). \tag{38}$$

The first term represents N samples of k least significant bits and is thus given by

$$\mathscr{L}(\tilde{L}_k^0) = Nk. \tag{39}$$

The second term, conveniently denoted by

$$F_k = \mathscr{L}(\psi_1[\tilde{M}_0^{n,k}]) \tag{40}$$

is simply the length of a fundamental sequence generated for the most significant n-k bit samples of $\tilde{M}_0^n$. We will investigate its relationships in the following paragraphs.

Calculation of F_k

Let m_j denote the jth sample of preprocessed data sequence $\tilde{M}_0^n$ so that

$$\tilde{M}_0^n = m_1 * m_2 * m_3 * \ldots * m_N. \tag{41}$$

Each sample of $\tilde{M}_0^n$ can be represented in standard binary form with n bits so that

$$m_j = b_j^{n-1} \cdot 2^{n-1} + b_j^{n-2} \cdot 2^{n-2} + \ldots + b_j^0 \tag{42}$$

where b_j^{n-1} is the most significant **bit** and b_j^0 is the least significant bit in this representation.

Using this notation and (14) we have

$$\begin{aligned}
F_0 &= N + \sum_{j=1}^{N} \sum_{\ell=0}^{n-1} b_j^\ell \cdot 2^\ell \\
F_1 &= N + \sum_{j=1}^{N} \sum_{\ell=1}^{n-1} b_j^\ell \cdot 2^{\ell-1} \\
&\quad \vdots \qquad\qquad \vdots \\
F_k &= N + \sum_{j=1}^{N} \sum_{\ell=k}^{n-1} b_j^\ell \cdot 2^{\ell-k}.
\end{aligned} \tag{43}$$

After some manipulation we can relate F_k to F_{k-1} by the expression

$$2F_k = F_{k-1} + N - \sum_{j=1}^{N} b_j^{k-1}. \tag{44}$$

Expanding (44), any F_k can be related to F_0 (fundamental sequence length for the original $\tilde{M}_0^n$) by

$$F_k \geq N + 2^{-k}\left(F_0 - N - \sum_{j=1}^{N}\sum_{\ell=0}^{k-1} b_j^{\ell} \cdot 2^{\ell}\right) \tag{45}$$

where we recognize the last term inside the parenthesis as the sum of all the k-bit least significant samples making up $\tilde{L}_k$. That is

$$F_k = \left\{\begin{array}{l}\text{Fundamental}\\ \text{sequence}\\ \text{Length for}\\ \tilde{M}_0^{n,k}\end{array}\right\} = N + 2^{-k}\left(F_0 - N - \left\{\begin{array}{l}\text{Sum of}\\ \text{k bit}\\ \text{samples}\\ \text{in } \tilde{L}_k^0\end{array}\right\}\right) \tag{46}$$

Substituting (45) or (46) and (39) back into (38) yields an exact calculation for $\mathscr{L}(\psi_{1,k}[\tilde{M}_0^n])$.

Simplified Calculations

Whereas (45) or (46) may reduce the implementation requirements for precise calculations of the $\{F_k\}$, significant further simplifications result if we are willing to accept an estimate of the $\{F_k\}$ based on F_0 and the assumption that the k least significant bits are completely random. It was the latter assumption that led us to the split sample modes in the first place. Then, if we assume for all j and $\ell \leq k$

$$b_j^{\ell} = \left\{\begin{array}{ll} 1 & \text{with probability } 1/2\\ 0 & \text{with probability } 1/2\end{array}\right. \tag{47}$$

and substitution in (45) yields

$$E\left\{F_k|F_0\right\} = \bar{F}_k \cong 2^{-k} F_0 + \frac{N}{2}(1-2^{-k}) \tag{48}$$

where $E\{\cdot\}$ denotes statistical expectation.

Adding Nk from (39) provides a practical estimate for $\mathscr{L}(\psi_{1,k}[\tilde{M}_0^n])$ in (38)

$$\mathscr{L}(\psi_{1,k}[\tilde{M}_0^n]) \approx \gamma_{1,k}(\tilde{M}_0^n) \equiv 2^{-k}F_0 + \frac{N}{2}(1-2^{-k}) + Nk. \tag{49}$$

The latter estimate and (48) are shown in a more convenient form in Table 3 for $k \leq 5$.

By (37) the selection of code operator $\psi_{1,k}[\cdot]$ can be accomplished by choosing the k for which $\gamma_{1,k}(\tilde{M}_0^n)$ is minimum. This results in the clear cut decision regions as shown in the rightmost column of Table 3.

Test results indicated that $\psi_{11}[\cdot]$ performance using either F_k or the simpler $\bar{F}_k$ in Table 3 could be expected to be almost identical. The largest observed performance difference for the eight-option code in (28) was 0.01 b/p.

FAST COMPRESSOR[10]

Consider now an extremely simple and hence computationally "FAST" $\psi_{11}[\cdot]$ based on the $\psi_{\phi,k}[\cdot]$ options in (24). By definition a $\psi_{\phi,k}[\cdot]$ code operator is simply to represent all of input sequence $\tilde{M}_0^n$ by $\tilde{L}_k$ (the sequence of k least significant bits). Since this process is reversible only if all the n − k most significant bits of $\tilde{M}_0^n$ are fixed (e.g., zero), we say that (using (42)) $\psi_{\phi,k}[\tilde{M}_0^n] = \tilde{L}_k^0$ is a valid noiseless representation if, for all j

$$b_j^{n-1} = b_j^{n-2} = \ldots = b_j^{n-k} = 0. \tag{50}$$

Table 3. $\bar{F}_k$ and $\gamma_{1,k}[\tilde{M}_0^n]$.

k	$\bar{F}_k$ (Eq. 48)	$\gamma_{1,k}(\tilde{M}_0^n)$ (Eq. 49)	DECISION REGION
0	F_0	F_0	$N \le F_0 \le \frac{5}{2}N$
1	$\frac{2F_0 + N}{4}$	$\frac{2F_0 + 5N}{4}$	$\frac{5}{2}N < F_0 < \frac{9}{2}N$
2	$\frac{2F_0 + 3N}{8}$	$\frac{2F_0 + 19N}{8}$	$\frac{9}{2}N < F_0 < \frac{17}{2}N$
3	$\frac{2F_0 + 7N}{16}$	$\frac{2F_0 + 55N}{16}$	$\frac{17}{2}N < F_0 < \frac{33}{2}N$
4	$\frac{2F_0 + 15N}{32}$	$\frac{2F_0 + 143N}{32}$	$\frac{33}{2}N < F_0 < \frac{65}{2}N$
5	$\frac{2F_0 + 31N}{64}$	$\frac{2F_0 + 351N}{64}$	$\frac{65}{2}N < F_0 < \frac{161}{2}N$ (for eight bit data)

Choosing between $\psi_{\phi,k}[\cdot]$ options using the optimum decision criteria in (35) means choosing the shortest $\tilde{L}_k^0$ such that (50) is true. Then letting†

$$k^* = \left\lfloor \log_2 (\max_j m_j) \right\rfloor \tag{51}$$

we choose k as the smallest of the allowed options such that $k \ge k^*$. If all k are allowed then $k = k^*$.

† $\lfloor x \rfloor$ means the smallest integer greater than or equal to x.

Performance

The performance of a FAST compressor using the eight options $\psi_{\phi,1}[\cdot]$, $\psi_{\phi,2}[\cdot]$, ..., $\psi_{\phi,8}[\cdot] = \psi_3[\cdot]$ and an input block of N = 16 is shown in Fig. 13. The graph indicates performance which remains uniformly about 0.6 bits/sample above the entropy line throughout the Voyager entropy range of interest. Its computational simplicity makes this algorithm the primary candidate for the Voyager Uranus encounter.†

FAST/FS

Exchanging $\psi_{\phi,1}[\cdot]$ for fundamental sequence operator $\psi_1[\cdot]$ in the above set of options leads to a noticeable improvement at lower entropies as illustrated in Fig. 13.

---

† Recent tests indicate that a FAST block size of N = 5 is the best choice for Voyager, providing an improved performance of 0.13 bits/sample.

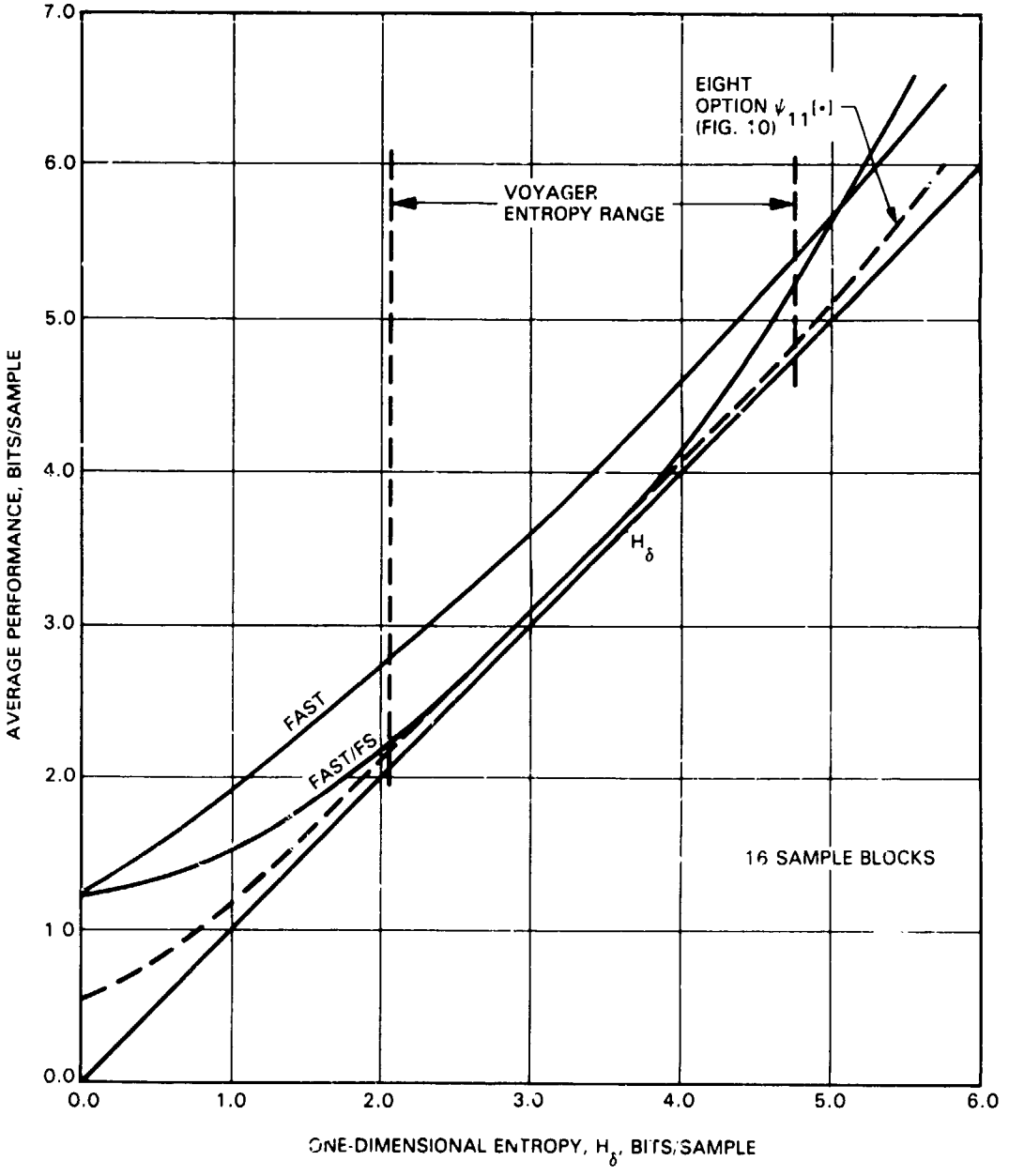

Fig. 13. FAST and FAST/FS Performance.

V. GENERALIZED SPLIT-SAMPLE MODES: CODING FOR SPIKES

The Voyager camera system generates reseau marks which appear as occasional, unusually large values of δ to any of the $\psi[\cdot]$ coding algorithms just discussed. These operators are generally protected against catastrophic behaviour by the inclusion of "back-up" operator $\psi_3[\cdot]$. Consequently the aggregate impact of these spikes averaged over an image is minor. However, the reseau marks are so arranged that an individual line will tend to have either several such spikes or none at all. Further, the buffering limitations and data format structure on Voyager require fixed line rates so that the effect of these events on an individual line cannot be averaged out over succeeding lines. This section investigates remedies to this type of situation.

The general problem is characterized by symbol probability distributions like that in Fig. 14.

Lower valued numbers exhibit the desired probability ordering in (1), $p_0 \geq p_1 \geq p_2 \geq \ldots$ but, due to perhaps an independent data source such as reseau marks or noise spikes, one or more very large numbers occur with significantly higher probability than some of the smaller numbers. Thus condition (1) is not well approximated for **all** the possible input numbers. This preprocessing problem is dealt with here by generalizing the split-sample modes to allow the two distinct distributions in Fig. 14 to be dealt with independently.

OPERATOR $SS_1^{n',k'}[\cdot]$

Consider now a second version of split-sample operations, $SS_1^{n',k'}[\cdot]$ shown in Fig. 15 where we use the parameters n' and k' to emphasize the distinction from previous definitions. The structure appears identical to that shown in Fig. 6 but the output sequences of least significant bit samples, $\tilde{L}_{k'}^{+}$, and most significant bit samples, $\tilde{M}_1^{n'\,k'}$, are defined differently.

Starting again with a preprocessed data sequence of n' bit samples denoted $\tilde{M}_0^{n'} = m_1 * m_2 * \ldots * m_N$ as in (41), let

$$j_1 < j_2 < j_3 < \ldots < j_L \tag{52}$$

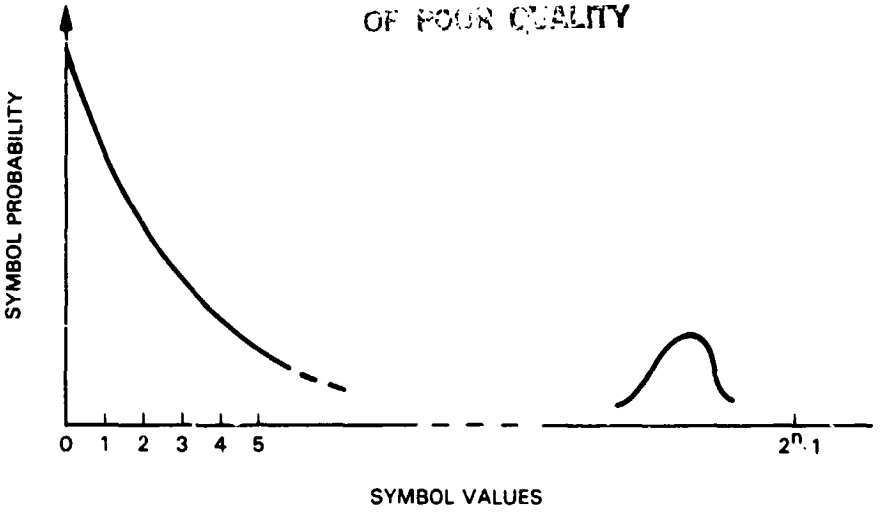

Fig. 14. Effect of Occasional Spikes.

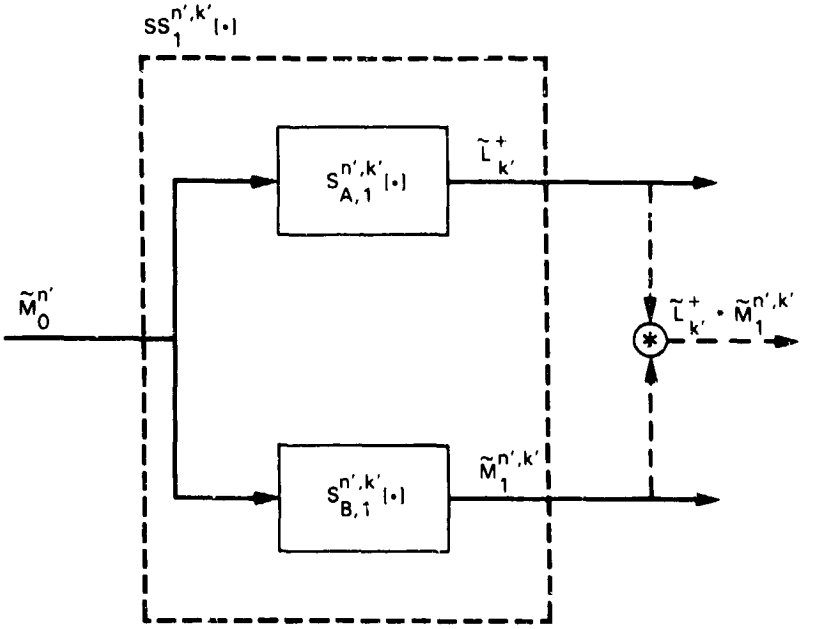

Fig. 15. Split-Sample Operation, $SS_1^{n',k'}[\cdot]$.

denote the location of all m_j samples such that

$$m_{j_\ell} \geq 2^{k'} - 1, \quad \ell = 1, 2, \ldots, L \tag{53}$$

and $0 \leq L \leq N$. Now concatenate, in order of occurrence, all the L samples of $\tilde{M}_0^{n'}$ which meet this condition and define the result by

$$S_{B,1}^{n',k'}[\tilde{M}_0^{n'}] = \tilde{M}_1^{n',k'} = m_{j_1} * m_{j_2} * \ldots * m_{j_L}. \tag{54}$$

Now for $j = 1, 2, \ldots, N$ let

$$m_j^+ = \begin{cases} 2^{k'} - 1 & \text{if } m_j \geq 2^{k'} - 1 \\ m_j & \text{Otherwise} \end{cases} \tag{55}$$

and form the sequence

$$S_{A,1}^{n',k'}[\tilde{M}_0^{n'}] = \tilde{L}_{k'}^{+} = m_1^+ * m_2^+ * \ldots m_N^+. \tag{56}$$

Limiting Cases

We note the limiting cases for $S_{B,1}^{n',k'}[\cdot]$ and $S_{A,1}^{n',k'}[\cdot]$ by

$$\begin{aligned} S_{B,1}^{n',n'}[\tilde{M}_0^{n'}] &= \tilde{M}_1^{n',n'} = \phi \text{ (i.e., Empty set)}^{\dagger} \\ S_{A,1}^{n',n'}[\tilde{M}_0^{n'}] &= \tilde{L}_{n'}^{+} = \tilde{M}_0^{n'} \end{aligned} \tag{57}$$

$^{\dagger}\tilde{M}_1^{n',n'}$ can actually be non-empty if and only if an m_j equals its maximum, $2^{n'} - 1$. In most real situations this would happen with very low probability so that for all practical purposes (57) is true.

and

$$S^{n',0}_{B,1}[\tilde{M}^{n'}_0] = \tilde{M}^{n',0}_1 = \tilde{M}^{n'}_0$$

$$S^{n',0}_{A,1}[\tilde{M}^{n'}_0] = \tilde{L}^{+}_0 = 0 * 0 * \ldots * 0. \tag{58}$$

Observe that $\tilde{M}^{n'}_0$ can be retrieved from the two sequences $\tilde{L}^{+}_k$, and $\tilde{M}^{n',k'}_1$. All samples of $\tilde{M}^{n'}_0$ which are less than $2^{k'} - 1$ appear unchanged in $\tilde{L}^{+}_k$, although they can be represented in fixed length form with only k' bits instead of the initial n'. Any sample in $\tilde{L}^{+}_k$, equal to $2^{k'} - 1$ is interpreted as a "flag" for a decoder to look for the true n' bit value in the $\tilde{M}^{n',k'}_1$ sequence **following** $\tilde{L}^{+}_{k'}$. Such n'-bit samples are unchanged from their values in $\tilde{M}^{n'}_0$ and appear in their original order of occurrence.

Example

To illustrate these operations in a less abstract manner, consider the N = 12 sample sequence

$$\tilde{M}^{n'}_0 = \tilde{M}^{8}_0 = 0, 1, 5, 3, 2, 1, 0, 3, 127, 1, 3, 0. \tag{59}$$

Applying operator $S^{8,3}_{A,1}[\cdot]$ we get, by (55) and (56), the sequence of 3-bit numbers

$$S^{8,3}_{A,1}[\cdot] = \tilde{L}^{+}_3 = 0, 1, 5, 3, 2, 1, 0, 3, 7, 1, 3, 0. \tag{60}$$

Observe that the ninth sample of $\tilde{M}^{8}_0$ has been reduced from 127 to 7. Only this same ninth sample satisfies (53) where $m_9 > 2^3 - 1 = 7$ so that by (54)

$$S^{8,3}_{B,1}[\tilde{M}^{8}_0] = \tilde{M}^{8,3}_1 = m_9 = \text{eight bit sample} = 127. \tag{61}$$

Further Processing of $\tilde{L}_{k'}^{+}$

The samples of $\tilde{L}_{k'}^{+}$ take on the values 0, 1, 2, ..., $2^{k'}-1$. Sample values less than $2^{k'}-1$ occur with the same probability as in $\tilde{M}_0^{n'}$ in Fig. 14 and so by assumption exhibit the desired ordering in (1). However, a sample of $\tilde{L}_{k'}^{+}$ equals $2^{k'}-1$ if any sample of $\tilde{M}_0^{n'}$ exceeds $2^{k'}-2$ and may thus occur more frequently than some of the smaller numbers. This is illustrated in Fig. 16.

To correct this situation we specify the mapping $\Omega_\alpha^{k'}[\cdot]$ of $\tilde{L}_{k'}^{+}$ into $\tilde{L}_{k'}^{++}$.

By (56), $\tilde{L}_k^{+} = m_1^{+} * m_2^{+} * \ldots m_N^{+}$ where the samples of $\tilde{L}_k^{+}$ are defined in (55). Then

$$\tilde{L}_k^{++} = \Omega_\alpha^{k'}[\tilde{L}_{k'}^{+}] = m_1^{++} * m_2^{++} * \ldots * m_N^{++} \tag{62}$$

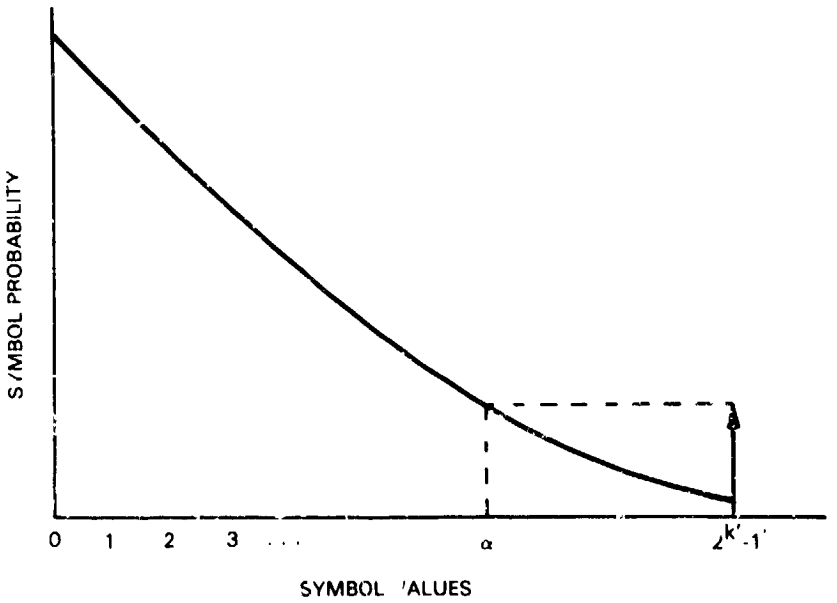

Fig. 16. Probability Distribution for Samples of $\tilde{L}_{k'}^{+}$.

where

$$m_j^{++} = \begin{cases} \alpha & \text{if } m_j^{+} = 2^{k'} - 1 \\ m_j^{+} & \text{if } m_j^{+} < \alpha \\ m_j^{+} + 1 & \text{if } \alpha \le m_j^{+} < 2^{k'} - 1 \end{cases} \tag{63}$$

and

$$\Omega_{k'}^{k'}[\tilde{L}_{k'}^{+}] = \tilde{L}_{k'}^{+}.$$

Optimally, α should be chosen such that

$$\Pr[m_j^{+} = 2^{k'} - 1] \begin{cases} \ge \Pr[m_j^{+} = i],\ i \ge \alpha \\ \\ \le \Pr[m_j^{+} = i],\ i \le \alpha \end{cases} \tag{64}$$

to ensure condition (1) for sequences of m_j^{++} (see Fig. 16). However, in practice such precise knowledge of data statistics is unlikely to be sufficient to ensure such optimality. In some cases an adaptive update of α may be desirable.

GENERALIZED SPLIT-SAMPLE CODER, $\psi_{i,k,k'}^{\alpha}[\cdot]$

Cascading the split-sample operations $SS_0^{n,k}[\cdot]$ and $SS_1^{n',k'}[\cdot]$ provides the structure for a generalized split-sample operation $SS^{n,k,k'}[\cdot]$ and an extended set of noiseless code operators, $\psi_{i,k,k'}^{\alpha}[\cdot]$. This structure is illustrated in Fig. 17. Starting from the left, n-bit/sample input sequence $\tilde{M}_0^n$ is first processed by the basic $SS_0^{n,k}[\cdot]$ operation (Fig. 8) to yield the sequence of k least significant bits, $\tilde{L}_k^0$, and the sequence of $n' = n - k$ most significant bits, $\tilde{M}_0^{n,k}$, which we relabel for further processing as

$$\tilde{M}_0^{n'} = \tilde{M}_0^{n,k}. \tag{65}$$

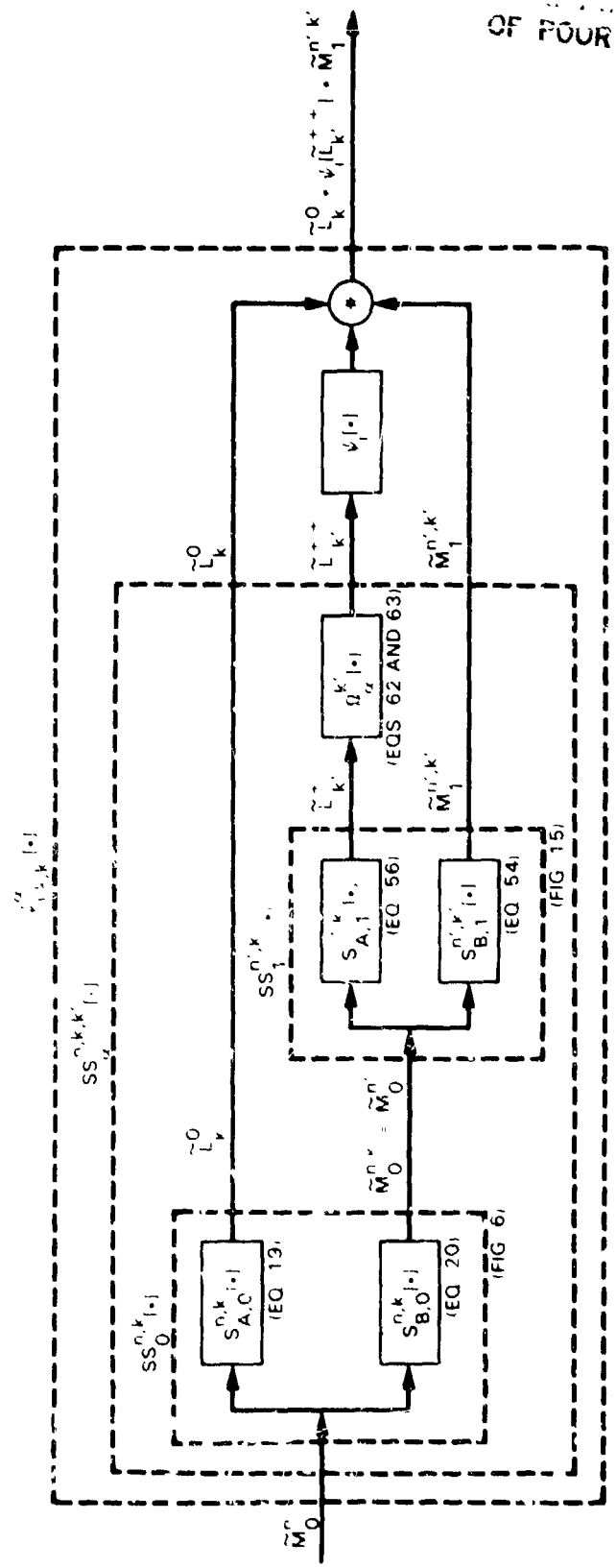

Fig. 17. Code Operator $\psi^{\alpha}_{i,k,k'}[\cdot,\cdot]$.

The most significant bit samples are then split by $SS_1^{n',k'}[\cdot]$ (Fig. 15) into a sequence of modified k' bit samples, $\tilde{L}_{k'}^{+}$, and a variable length sequence of n' bit samples $\tilde{M}_1^{n',k'}$ $\tilde{L}_{k'}^{+}$, is then further processed by $\Omega_\alpha^{k'}[\cdot]$ (62) into $\tilde{L}_{k'}^{+\,+}$. Then we have

$$SS_\alpha^{n,k,k'}[\tilde{M}_0^n] = \left\{\tilde{L}_k^0, \tilde{L}_{k'}^{+\,+}, \tilde{M}_1^{n',k'}\right\}. \tag{66}$$

An extended class of noiseless coders for $\tilde{M}_0^n$ is defined by

$$\psi_{i,k,k'}^{\alpha}[\tilde{M}_0^n] = \tilde{L}_k^0 * \psi_i[\tilde{L}_{k'}^{+\,+}] * \tilde{M}_1^{n',k'} \tag{67}$$

where $\psi_i[\cdot]$ is any previously defined code operator.†

APPLICATION OF $\psi_{i,k,k'}^{\alpha}[\cdot]$ TO VOYAGER

Unlike random noise spikes, the effect of periodically placed reseau marks on preprocessed $\tilde{M}_0^n$ distributions is both infrequent and local in nature. That is, only those lines containing reseau marks will exhibit unusually high frequencies of occurrence of very large symbol values. More locally, this effect appears only in regions within lines where reseau marks occur. Statistically a Voyager image data source then appears as a normal image source for very long periods (lines where no reseaus appear) interrupted periodically by stretches (several lines) where statistical characteristics alternate between normal image data and image data containing spikes (see Fig. 14).

General Operator Definitions

First we extend the definition of $\psi_{11}[\cdot]$ to handle sequences of data blocks. Let $\tilde{Z} = \tilde{Z}_1 * \tilde{Z}_2 * \ldots * \tilde{Z}_T$ be a sequence of T preprocessed data blocks (i.e., any $\tilde{Z}_i$ can be interpreted to be an $\tilde{M}_0^n$) and define $\psi_{12}[\cdot]$ by

$$\psi_{12}[\tilde{Z}] = \psi_{11}[\tilde{Z}_1] * \psi_{11}[\tilde{Z}_2] * \ldots * \psi_{11}[\tilde{Z}_T]. \tag{68}$$

---

†Observe that some applications might benefit from additional coding of $\tilde{M}_1^{n',k'}$.

$\psi_{12}[\bullet]$ is the operation of applying a **specific** $\psi_{11}[\bullet]$ operator individually to each $\tilde{Z}_i$ and then concatenating the results in order of occurrence.† $\psi_{12}[\tilde{Z}]$ might specify the coding of a complete line of image data for example.

Now let

$$\psi^a_{11}[\bullet] \tag{69}$$

be a $\psi_{11}[\bullet]$ code operator (see 27) selected to be used on data which **does not contain spikes** (e.g., the eight option $\psi_{11}[\bullet]$ in 28). Then by (68) the corresponding operator for a sequence of many data blocks such as a complete line is

$$\psi^a_{12}[\bullet]. \tag{70}$$

Next, let

$$\psi^b_{11}[\bullet] \tag{71}$$

be an operator formed from one or more of the $\psi^{\alpha}_{i,k,k'}[\bullet]$ options which are chosen to work effectively when **data spikes are present.**

Now form a 2-option $\psi_{11}[\bullet]$ called $\psi^c_{11}[\bullet]$ which chooses between $\psi^a_{11}[\bullet]$ **and** $\psi^b_{11}[\bullet]$. $\psi^c_{11}[\tilde{M}^n_0]$ takes the form

$$\psi^c_{11}[\tilde{M}^n_0] = \begin{cases} 0 * \psi^a_{11}[\tilde{M}^n_0] \\ \\ 1 * \psi^b_{11}[\tilde{M}^n_0] \end{cases} \tag{72}$$

where the '0' or '1' identifies to a decoder whether $\psi^a_{11}[\bullet]$ or $\psi^b_{11}[\bullet]$ respectively, was used to represent $\tilde{M}^n_0$.

---

†This is much the same as the extension of $\psi_4[\bullet]$ to $\psi_5[\bullet]$ and $\psi_6[\bullet]$ in Ref. 1.

Again by (68), the corresponding multiple block or full-line (for imaging) application of $\psi^c_{11}[\cdot]$ follows as

$$\psi^c_{12}[\cdot]. \tag{73}$$

Finally, define operator

$$\psi^d_{11}[\cdot] \tag{74}$$

as a two-option $\psi_{11}[\cdot]$ which chooses between $\psi^a_{12}[\cdot]$ and $\psi^c_{12}[\cdot]$. For imaging this decision is made on a line-by-line basis. $\psi^d_{11}[\cdot]$ allows the additional 1-bit per block overhead of $\psi^c_{11}[\cdot]$ to be avoided when there are long stretches of data without spikes (i.e., the lines between reseau marks in imaging).

Split-FS Options for Voyager

For the Voyager imaging application we take $\psi^a_{11}[\cdot]$ as made up of

$$\{\text{eight-options in (28)}\} \tag{75}$$

and based on experimental evidence of symbol distributions in the regions where reseaus occur, the assumed eight-option set forming $\psi^b_{11}[\cdot]$ is $\psi^\alpha_{i,k,k'}[\cdot]$ with

$$i = 1, \alpha = 1 \tag{76a}$$

and the $\{k,k'\}$ pairs

$$\{0,2\}, \{0,3\}, \{0,4\}, \{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}. \tag{76b}$$

Performance. Figure 18 illustrates the impact of $\psi^d_{11}[\cdot]$ coding on two images of distinctly different entropies from the Voyager test set in Fig. 5. The originals of these images are shown at the top and the results of applying both $\psi^a_{12}[\cdot]$ and $\psi^d_{11}[\cdot]$ under a fixed line rate constraint of 3.0 b/p are shown directly below. The white areas on the right side of each image represent pixels that are missing because the allotment of (3.0)(800) bits per line had been used up before the complete line could be transmitted.

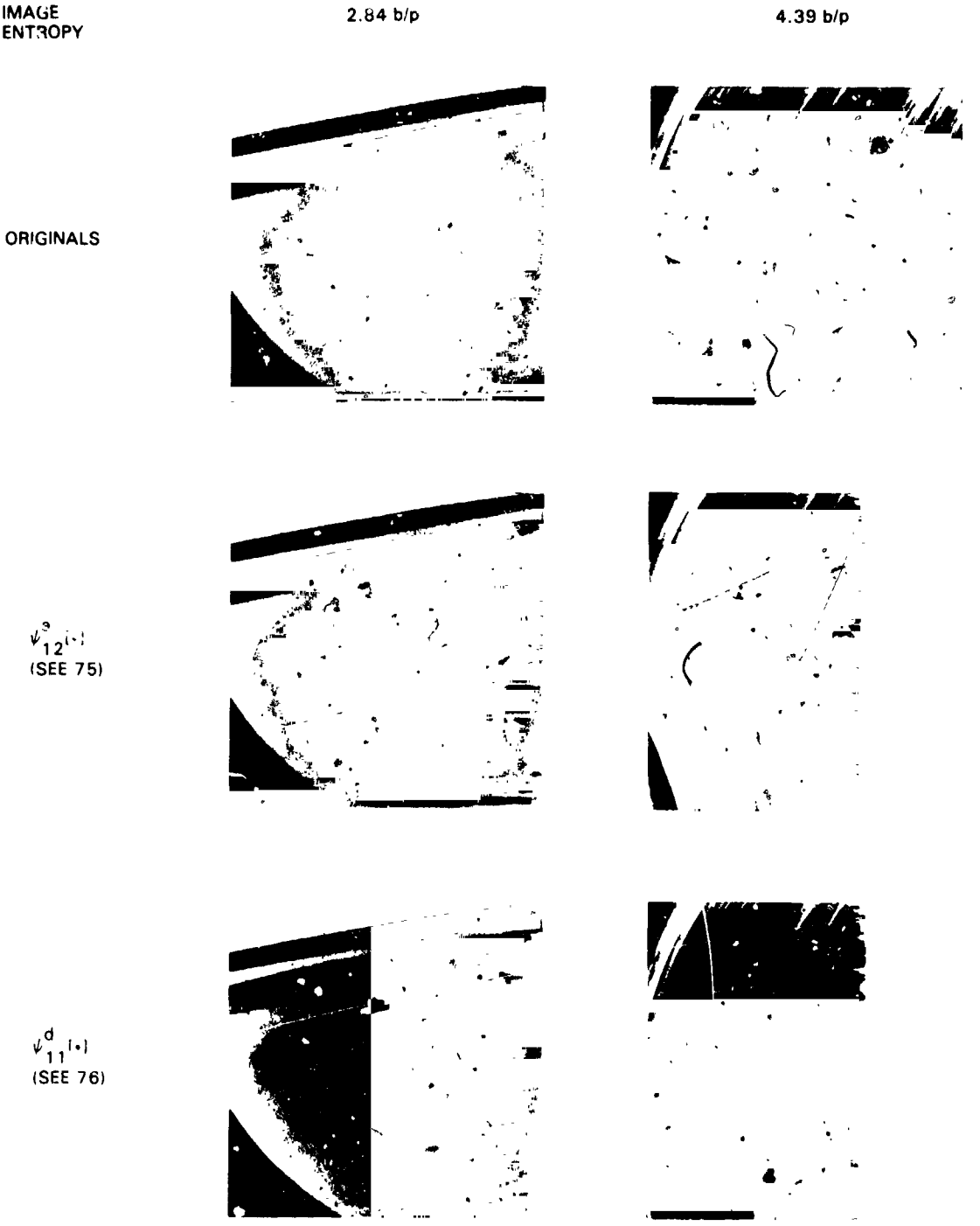

Fig. 18. Comparison of Pe[illegible]ts Using $\psi^a_{12}[\cdot]$ and $\psi^d_{11}[\cdot]$ with [illegible]ate 3.0 b/p.

Under such a constraint the effect of reseau marks is to substantially increase the number of missing pixels. A comparison of the images generated using $\psi^{a}_{12}[\cdot]$ and $\psi^{d}_{11}[\cdot]$ reveals that $\psi^{d}_{11}[\cdot]$ reduces the number of pixels caused by reseau marks by approximately 50% with performance on non-reseau lines unchanged.

A more detailed look at performance showed that $\psi^{d}_{11}[\cdot]$ improved performance on the lower entropy image by:

a) as much as 4 b/p, and an average of 1 b/p on individual **16 sample blocks** containing reseau marks;

b) as much as 0.3 b/p, and an average of 0.2 b/p on **lines** containing reseau marks.

c) an average of 0.04 b/p over a complete image

Absolute gains on the higher entropy image were smaller.

Removing noiseless constraint. A slight modification to the detailed definition of $\psi^{\alpha}_{i,k,k'}[\cdot]$ in Fig. 17 can permit the flagged samples of $\tilde{M}^{n',k'}_{1}$ to be reduced in quantization. The result is to eliminate completely the remaining reseau-caused additional loss of data in Fig. 18. The necessary penalty is the acceptance of local quantization error on the reseau marks themselves.

Decision process for $\psi^{d}_{11}[\cdot]$. With the options chosen in (75) and (76) the $\psi^{d}_{11}[\cdot]$ decision process can be accomplished by coding a line with $\psi^{c}_{11}[\cdot]$ and saving the 1-bit block identifiers (see 72) as a single line header. If all block ID's are zero they can be discarded, since in this case, the line has been coded using only $\psi^{a}_{11}[\cdot]$.

FAST Compressor for Spikes

We now investigate computationally simpler options for $\psi^{a}_{11}[\cdot]$, $\psi^{b}_{11}[\cdot]$, etc.

Let $\psi^{a}_{11}[\cdot]$ of (69) be

$$\{\text{The eight-option FAST compressor in (24), (50) and (51)}\} \qquad (77)$$

and for an eight-option $\psi^{b}_{11}[\cdot]$ in (71) we use $\psi^{\alpha}_{i,k,k'}$ with $k = 0$ and $\alpha = 2^{k'} - 1$ and for various k' we choose i to denote selected (FAST Compressor) code options from (24). These choices greatly simplify the general diagram for $\psi^{\alpha}_{i,k,k'}[\cdot]$ in Fig. 17. With $k = 0$, the basic split-sample operation $SS^{n,k}_{0}[\cdot]$ is not performed so that $\tilde{M}^{n}_{0}$ directly enters $SS^{n',k'}_{1}[\cdot]$ with $n = n'$. Since $\alpha = 2^{k'} - 1$, the $\Omega^{k'}_{\alpha}[\cdot]$ mapping is not performed and simply passes its input on unchanged. The resulting structure is shown in Fig. 19 where we have replaced $\psi_{i}[\cdot]$ with its designated form from (24), $\psi_{\phi,k''}[\cdot]$, with new parameter k''. Our eight-option $\psi^{b}_{11}[\cdot]$ is then defined by the following $\{k',k''\}$ pairs

$$\{2,1\},\ \{2,2\},\ \{2,3\},\ \{2,4\} \tag{78a}$$

and

$$\{3,1\},\ \{3,2\},\ \{3,3\},\ \{3,4\}. \tag{78b}$$

Performance. A simplified $\psi^{d}_{11}[\cdot]$ code operator in (74) is obtained by substituting these simplified algorithms. Fig. 20 illustrates the result of this substitution. Using the same two images as in Fig. 18, the impact of this "modified FAST compressor" on reseau effects is displayed. Again, a fixed line rate of 3.0 b/p is assumed. The results of $\psi^{d}_{11}[\cdot]$ from Fig. 18 appear at the top as a reference instead of the original images.

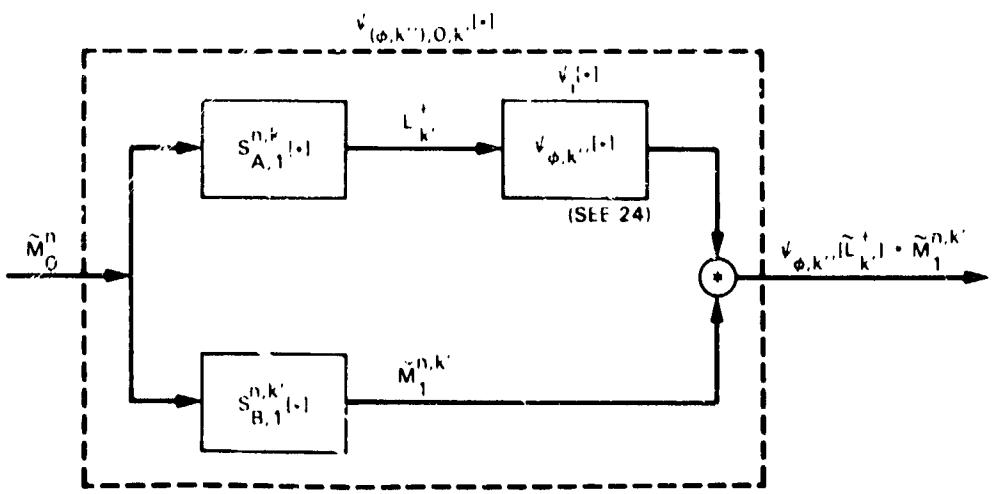

Fig. 19. FAST Options for $\psi^{b}_{11}[\cdot]$.

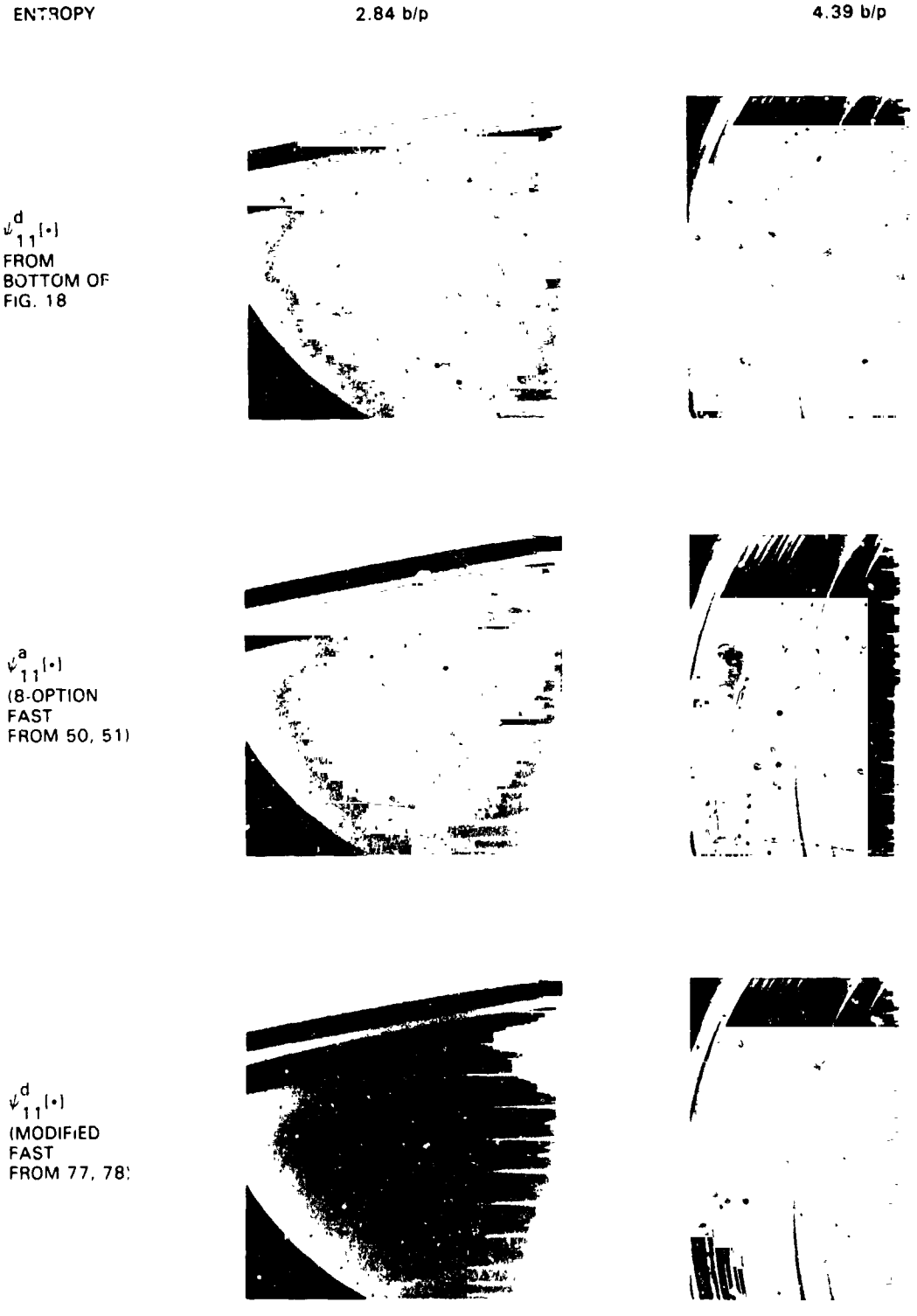

Fig. 20. Comparison of Reseau Effects Using 8-Option FAST and Modified FAST with Fixed Line Rate of 3.0 b/p.

The modified FAST $\psi^{d}_{11}[\bullet]$ reduced the number of reseau-caused missing pixels by about ⅓ on both images. This is a lower percentage reduction than observed in Fig. 18 with the more complex $\psi^{d}_{11}[\bullet]$. This is because the reseau effects on the basic FAST algorithm ($\psi^{a}_{11}[\bullet]$ here) are more pronounced. The absolute gains are actually very close.

As with the more complex $\psi^{d}_{11}[\bullet]$, a removal of the noiseless constraint on reseau regions could allow further reductions in reseau effects.

VI. ADDITIONAL IMPROVEMENTS FOR IMAGE DATA AND PERFORMANCE SUMMARY

The test data used to evaluate algorithm performance in Sections II-V was exclusively derived from image data sources. However, by choosing to follow the optional preprocessing Structure A in Fig. 17 all **performance vs. entropy** results are generally applicable to data sources which can be made to appear locally memoryless and have symbol probability distributions approximating condition (1) or the unusual situation caused by noise spikes depicted in Fig. 14. Here we note improvements in absolute performance directly attributable to simple modifications of the basic "image" preprocessing functions: a) prediction, and b) the mapping of prediction error into the positive integers.

TWO-DIMENSIONAL PREDICTOR

All results generated thus far have been based on the basic one-dimensional prediction algorithm (and corresponding one-dimensional entropy) defined in Table 1. Simply replacing that predictor with the two-dimensional version in the same table can yield substantial gains in **absolute** performance at the higher entropy values as illustrated in Fig. 21.[4] The graph plots two-dimensional entropy vs. one-dimensional entropy for the same image test set we have used throughout. Entropy reductions of as much as 0.6 b/p can be expected.

Coding performance **relative** to data entropy is independent of the predictor used since all the desired conditions for preprocessing are met in either case. Hence, **absolute** performance of a given coding algorithm can be expected to follow any reductions in entropy resulting from the use of two-dimensional prediction. Figure 21 thus depicts the real gains in performance which can be expected to be passed on to such algorithms as $\psi_{11}[\cdot]$ in Fig. 10 or the FAST Compressor in Fig. 13.

Adaptive Prediction

The graphs in Fig. 21 display average results for complete images. In Voyager's case we found some unusual quirks in the test set which occasionally made the one-dimensional predictor significantly better than the two-dimensional predictor. Choosing between the two options on a line-by-line basis eliminated any problems. Further sophistication seemed unwarranted.

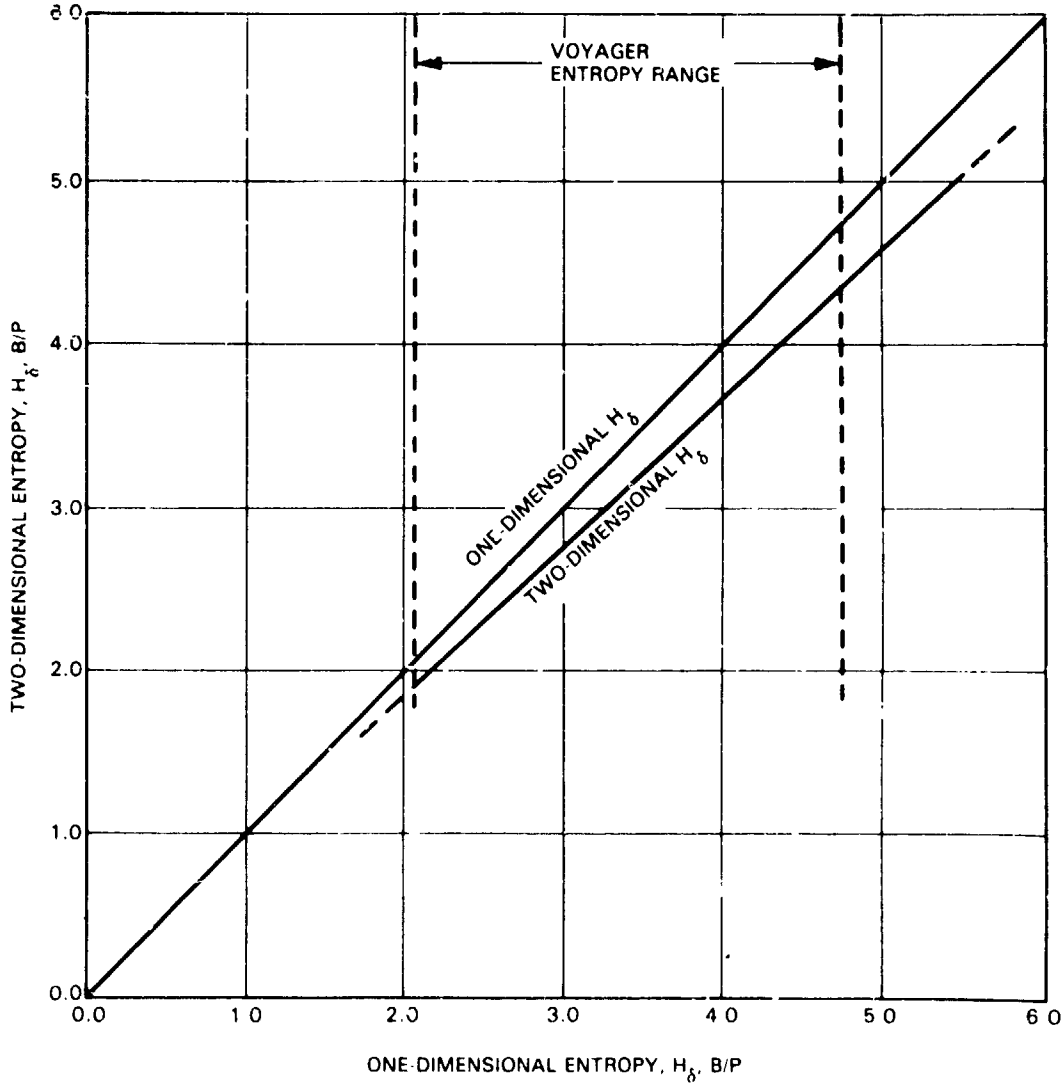

Fig. 21. Two-Dimensional vs. One-Dimensional Entropy.

Pictorial Results

The results of combining the generalized split-sample modes of Section V with two-dimensional adaptive prediction are illustrated in Fig. 22. For comparison purposes pictures from both Figs. 18 and 20 have been reproduced here. The top two images have been transferred from the middle row of images in Fig. 20 and represent the results of the basic FAST compressor using one-dimensional (1-D) prediction. Recall that this is the algorithm currently implemented for the Voyager Uranus encounter. The second row

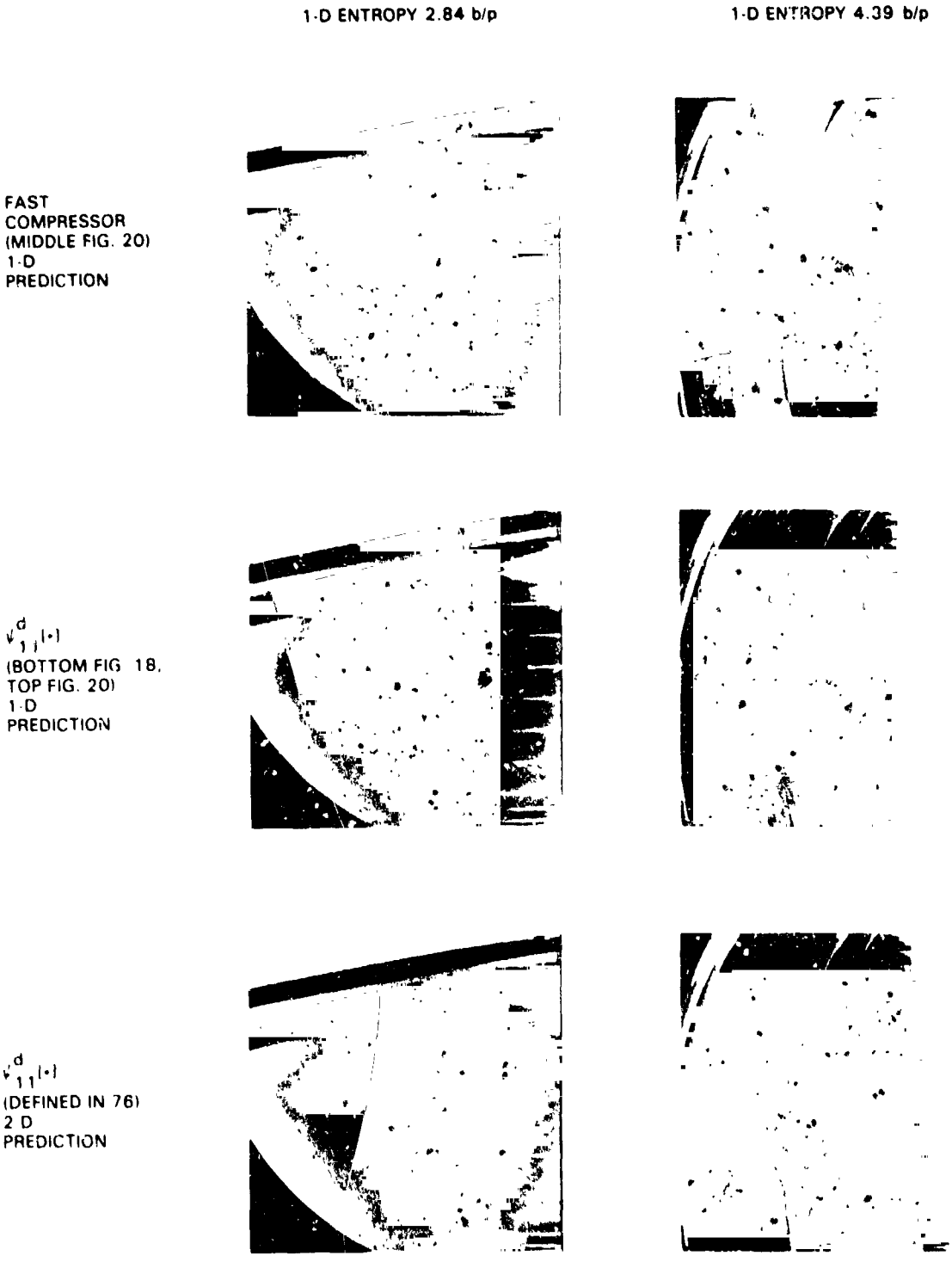

Fig. 22. Impact of 2-D Prediction.

of images has been transferred from the bottom of Fig. 18 (or the top of Fig. 20) and is the result of using the $\psi^{d}_{11}[\cdot]$ defined in (76) with one-dimensional prediction. The bottom pair of images represent the results of using this same operator in combination with two-dimensional (adaptive) prediction (2-D). The advantages of two-dimensional prediction are quite apparent, particularly on the high entropy image.

IMPROVEMENTS FROM DYNAMIC RANGE CONSTRAINTS

All prior performance results were based on the basic mapping of prediction errors, Δ, into the integers, δ in (3). Replacing (3) with the more complex form in (6) takes into account dynamic range constraints. Test using both one and two-dimensional predictors, indicated typical performance improvements of from 0.01 to 0.03 b/p. There is no coding penalty in utilizing (6) under any condition.

REFERENCES

1. R. F Rice, "Some practical universal noiseless coding techniques," **JPL Publication 79-22**. Jet Propulsion Laboratory, Pasadena, CA, March 15, 1979.

2. R. F. Rice, "Practical universal noiseless coding," **1979 SPIE Symposium Proceedings**, Vol. 207, San Diego, CA, August 1979.

3. R. F. Rice, A. Schlutsmeyer, "Software for universal noiseless coding," **Proceedings of the 1981 International Conference on Communications**, Denver, Colorado, June 1981.

4. R. F. Rice, J. R. Plaunt, "Adaptive variable length coding for efficient compression of spacecraft television data," IEEE Trans. Commun. Technol., Vol. COM-19, part I, Dec. 1971, pp. 889-897.

5. R. F. Rice, "An advanced imaging communication system for planetary exploration," **1975 SPIE Symposium Proceedings**, Vol. 66, San Diego, CA, Aug. 21-22, 1975, pp. 70-89.

6. R. F. Rice, A. P. Schlutsmeyer, "Data compression for NOAA weather satellite systems," **1980 SPIE Symposium Proceedings**, Vol. 249, San Diego, CA, July 1980.

7. R. F. Rice et al, "Block adaptive rate controlled image data compression," **Proceedings of 1979 National Telecommunications Conference**, Washington, D.C., Nov. 1979.

8. R. F. Rice, "End-to-End imaging information rate advantages of various alternative communication systems," **JPL Publication 82-61**. Jet Propulsion Laboratory, Pasadena, CA, Sept. 1, 1982.

9. R. F. Rice, "Channel coding and data compression system considerations for efficient communication of planetary imaging data," **Technical Memorandum 33-695**. Jet Propulsion Laboratory, Pasadena, CA, June 15, 1974.

10. Options of "FAST" compressor proposed by Richard J. Rice.